Technische Universität Braunschweig

ISF

Projektarbeit

Entwicklung und Modellierung von Evolutionsszenarien für Delta-orientierte Softwareproduktlinien

Sophia Nahrendorf

13.12.2016

Institut für Softwaretechnik und Fahrzeuginformatik
an der
Technischen Universität Carolo-Wilhelmina in Braunschweig

Prof. Dr. Ina Schaefer

Sascha Lity, M.Sc.

Zusammenfassung

Softwareproduktlinien bieten aufgrund ihrer Variabilität eine hohe Variantenvielfalt an unterschiedlichen, resultierenden Produkten. Dies und das hohe Maß an Wiederverwendung von Softwarebausteinen innerhalb der Produktlinie sowie deren vielfältige Abhängigkeiten untereinander führen dazu, dass die Evolution von Softwareproduktlinien überaus komplex ist. Damit Inkonsistenzen und Fehler als Folge von Evolution weitestgehend vermieden werden können, müssen vorher die genauen Auswirkungen der geplanten Änderungen eingeschätzt werden können. Dies wird durch die eingehende Betrachtung und Untersuchung der Veränderungen in Form von Modellen erreicht. Daher sind Modellierungsmethoden notwendig, die eine übersichtliche und verständliche Darstellung bieten und gleichzeitig eine effiziente Analyse der Auswirkungen erlauben. Doch auch die vielzählig vorhandenen Ansätze, Evolution von Softwareproduktlinien zu modellieren, müssen ausgiebig erprobt und evaluiert werden, damit Fehler im Konzept ausgeschlossen werden können und eine zuverlässige und alltagstaugliche Anwendung nachgewiesen werden kann. Für diese Evaluation werden Fallstudien mit Evolutionshistorie im Kontext von Softwareproduktlinien benötigt, welche bisher allerdings nicht in ausreichendem Maß vorhanden sind. Daher werden in dieser Projektarbeit vier verschiedene Fallstudien vorgestellt, die Softwareproduktlinien beschreiben. Es handelt sich um einen Verkaufsautomaten, eine Scheibenwischanlage, eine Minenpumpanlage und ein Body Comfort System. Für diese Fallstudien werden Szenarien entwickelt, welche evolutionäre Veränderungen an den zugrundeliegenden Softwareproduktlinien illustrieren. Da es sich um delta-orientierte Fallstudien handelt, werden die Ansätze „Delta-Modellierung“ zur Darstellung von Variabilität und „Higher-Order Delta-Modellierung“ zur Modellierung von Evolution verwendet, um Modelle für die Fallstudien und Evolutionsszenarien zu erstellen. Die entwickelten Evolutionsszenarien werden in einem Eclipse-basierten Prototyp umgesetzt und in dieser Arbeit dokumentiert.

Stichwörter Softwareproduktlinien, Delta-Modellierung, Higher-Order Delta-Modellierung, Evolution

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Softwaresysteme unterliegen ständiger Evolution, sei es durch sich wandelnde Ansprüche der Kunden, durch Konkurrenz, die zu Innovationen zwingt, durch die Notwendigkeit, Fehler zu beheben oder Funktionen zu verbessern [24]. Um die Evolution möglichst effizient und fehlerfrei zu gestalten, unterliegt diese strengen Richtlinien der Softwareentwicklung. Ein wesentlicher Teil des Evolutionsprozesses sind hierbei Dokumentation und die Modellierung des Systems. Modelle helfen einen Überblick über das System zu bewahren, frühzeitig Fehler oder Widersprüchlichkeiten ausfindig zu machen, und gewährleisten eine bessere und eindeutigere Verständigung und Kommunikation zwischen den Stakeholdern, die am Entwicklungsprozess beteiligt sind [3].

Softwareproduktlinien [29] (SPL) sind aufgrund ihrer Variantenvielfalt besonders häufig der Evolution ausgesetzt [24]. SPLs beschreiben im Wesentlichen ähnliche Softwareprodukte der gleichen Anwendungsdomäne, die aus einem Kern an gemeinsamen Artefakten bestehen, jedoch zugleich unterschiedliche, individuelle Ausprägungen durch variable Artefakte besitzen. Artefakte sind Elemente des Entwicklungsprozesses, wie etwa Anforderungen, Modelle oder sogar Quellcode. Hierbei können Einschränkungen bestehen, welche dieser variablen Bausteine miteinander in einem Produkt zusammen eingesetzt werden dürfen, oder Bedingungen, welche von diesen zwangsläufig zusammen verwendet werden müssen. In einer SPL werden sowohl die gemeinsamen als auch die variablen Elemente implementiert und konkrete Produkte daraus abgeleitet, indem die in der SPL implementierten Artefakte wiederverwendet werden. Durch die Variabilität und die Wiederverwendung von Artefakten ist die Evolution von SPLs sehr komplex und wird daher teilweise bereits jahrelang im Voraus geplant, um Fehler zu vermeiden [24]. Veränderungen an den Kernelementen wirken sich auf alle abgeleiteten Produkte der SPL aus und Änderungen, die in direkter Beziehung mit anderen Elementen stehen, haben ebenfalls Auswirkungen auf jene. Daher muss die Evolution immer besonders mit Bedacht vorgenommen werden, um Inkonsistenzen in der kompletten SPL zu vermeiden.

Dementsprechend wichtig ist daher auch die Modellierung im Evolutionsprozess, um Inkonsistenzen und Fehler zu identifizieren [21, 24, 32]. Die Modellierung von SPLs ist äußerst komplex, da Modelle schnell sehr groß und unübersichtlich werden können, wenn versucht wird, die unterschiedlichen möglichen Produktvarianten einer SPL sowie die Beziehungen zwischen Komponenten, die sich gegenseitig bedingen oder ausschließen, darzustellen. Entsprechend schwieriger wird das Ganze noch, wenn immer wieder Veränderungen hinzukommen.

Es existieren verschiedene bewährte Ansätze SPLs zu modellieren [24, 32], unter anderem Delta-Modellierung [7, 30]. Bei der Delta-Modellierung wird eine Familie von Produkten durch ein ausgewiesenes Kernprodukt und eine Menge von Deltas dargestellt [13]. Die Deltas beschreiben dabei Abfolgen von Änderungen (Operationen) am Kernprodukt, wie Hinzufügen, Entfernen oder Modifizieren von Elementen, wodurch weitere Produktvarianten entstehen. Schwieriger sieht es bei der Modellierung von Evolution von SPLs aus, doch gibt es auch hier Ansätze [21, 23, 24]. Bei der Higher-Order Delta-Modellierung [21] wird etwa Evolution beschrieben, indem sogenannte Higher-Order

Deltas auf bestehende Delta-Modelle angewendet werden. So werden dann etwa neue Deltas zum Delta-Modell hinzugefügt, überflüssig gewordene Deltas entfernt oder bestehende Deltas modifiziert. Beim Modifizieren von Deltas wird die durch ein Delta beschriebene Abfolge von Operationen verändert, indem Operationen entfernt, modifiziert oder hinzugefügt werden. Jedoch müssen auch diese Ansätze evaluiert werden, um Fehler auszubessern, die korrekte Funktion zu zeigen und die Alltagstauglichkeit zu demonstrieren. Um neue Ansätze zur Modellierung von Evolution in SPLs zu evaluieren, ist es notwendig, dass genügend Fallstudien mit Evolutionshistorie zur Verfügung stehen, an denen diese Ansätze evaluiert werden können. Bisher existiert zu den im Kontext SPL verwendeten Fallstudien jedoch noch keine Historie in Form von Evolutionsszenarien.

Zu diesem Zweck sollen in dieser Arbeit - nach einem Überblick über die Themen Evolution, Softwareproduktlinien und insbesondere Evolution von Softwareproduktlinien – vier verschiedene Fallstudien auf mögliche Evolutionsszenarien hin untersucht werden. Das bedeutet, zu jeder dieser vier Fallstudien sollen entsprechend der Möglichkeiten mehrere Evolutionsszenarien entwickelt werden, die die unterschiedlichen Operationen Hinzufügen, Entfernen und Modifizieren von Deltas abdecken. Die zu betrachtenden Fallstudien sind in diesem Fall das Body Comfort System eines Fahrzeugs [22], eine Scheibenwischanlage, ein Verkaufsautomat und ein Minenpumpsystem [8]. Die zu den verschiedenen Fallstudien entwickelten Evolutionsszenarien sollen zunächst beschrieben werden. Des Weiteren soll dann ein bestehender Ansatz der Modellierung auf jedes Evolutionsszenario angewandt werden, um die verschiedenen Szenarien zu modellieren. Der Ansatz, der hierbei verwendet werden soll, ist Higher-Order Delta-Modellierung [21]. Zur Modellierung der Evolutionsszenarien soll der von Lity et al. [21] beschriebene, bestehende Eclipse-basierte Prototyp angewandt werden. Der Prototyp ist während der Verwendung zusätzlich hinsichtlich Bedienbarkeit und Funktionalität zu prüfen und bei Problemen gegebenenfalls anzupassen. Diese so entwickelten Evolutionsszenarien könnten in Zukunft genutzt werden, um weitere Ansätze im Kontext von SPL Evolution zu evaluieren.

Zusammenfassend werden in dieser Projektarbeit daher die folgenden Aufgaben bearbeitet:

- Einarbeitung in relevante Literatur zur Evolution von Softwareproduktlinien, Higher-Order Delta-Modellierung und die gegebenen Fallstudien
- Entwicklung von realitätsnahen Evolutionsszenarien für die Fallstudien
- Werkzeuggestützte Modellierung der entwickelten Evolutionsszenarien gemäß Higher-Order Delta-Modellierung
- Dokumentation der Ergebnisse

Strukturierung der Arbeit

In Kapitel 2 werden zunächst die Grundlagen über Softwareproduktlinien und Evolution von Softwareproduktlinien erläutert. Außerdem wird in Kapitel 2.2 Delta-Modellierung als Methode zur Modellierung von Variabilität in Softwareproduktlinien vorgestellt. Weiterhin wird in Kapitel 2.4 Higher-Order Delta-Modellierung als Möglichkeit zur Modellierung von Evolution von Softwareproduktlinien eingeführt. Die verwendeten Fallstudien Body Comfort System, Verkaufsautomat,

Scheibenwischanlage und Minenpumpanlage werden in Kapitel 3 vorgestellt. Für jede Fallstudie werden dafür Aufbau und Verhalten beschrieben und eine Ausgangssituation für die weitere Entwicklung von Evolutionsszenarien geschaffen. In Kapitel 4 werden, gegliedert nach Fallstudien, Evolutionsszenarien zu den Fallstudien entwickelt und modelliert. Nach einem Überblick über die Ausgangssituation werden die gewünschten Veränderungen beschrieben und grafisch veranschaulicht. Abschließend werden in Kapitel 5 die Ergebnisse der Arbeit zusammengefasst und ein abschließendes Fazit sowie ein Ausblick auf zukünftige Arbeiten gegeben.

2 Grundlagen

In diesem Kapitel werden die benötigten Grundlagen zum Verständnis dieser Arbeit erläutert. Dazu wird im Folgenden zuerst auf *Softwareproduktlinien* [29] näher eingegangen. Daraufhin wird *Delta-Modellierung* als Möglichkeit zur Modellierung von Softwareproduktlinien vorgestellt. Des Weiteren wird das Prinzip der *Evolution* von Softwareproduktlinien geschildert und *Higher-Order Delta-Modellierung* als Ansatz zur Modellierung von Evolution in Softwareproduktlinien beschrieben.

2.1. Softwareproduktlinien

Laut Clements und Northrop [9] ist eine Softwareproduktlinie eine Reihe von Software-intensiven Systemen, die sich gemeinsame Merkmale (engl. *features*) teilen und dadurch spezifische Ansprüche an eine bestimmte Marktnische erfüllen. Hierbei gilt, dass diese Systeme aus einem gemeinsamen Pool an *Softwarebausteinen* (engl. *assets*) auf eine vorgeschriebene Weise entwickelt werden. Solche Softwarebausteine können etwa Anforderungen, Domainmodelle, Softwarearchitekturen, Testfälle, Komponenten, Quellcode oder viele weitere Artefakte des Softwareentwicklungsprozesses sein. Der wesentliche Punkt einer Softwareproduktlinie ist dabei, dass diese Softwarebausteine explizit dafür entworfen wurden, wiederverwendet zu werden. Das Ziel einer SPL liegt darin, Massenproduktion (engl. *mass customization*) mit der Erfüllung indviueller Kundenwünsche zu kombinieren. Dies gelingt durch die individuelle Gestaltung eines Produktes auf Basis der Variabilitätsmöglichkeiten der Produktlinie.

Der Entwicklungsprozess von Softwareproduktlinien und deren Produkten ist nach Pohl et al. [29] in zwei Phasen unterteilt: *Domain Engineering* und *Application Engineering*. Die Phasen und ihre Teilphasen sind in der entsprechend Pohl et al. [29] nachgebildeten Abbildung 2.1 zu sehen.

Domain Engineering wird hierbei auch *Core Asset Development* [9] genannt und beschreibt das Erstellen von wiederverwendbaren Kernbausteinen sowie das Definieren von Gemeinsamkeiten und Variabilität der gesamten Softwareproduktlinie [29]. Dabei werden die Kernbausteine nicht nur konzeptionell entworfen, sondern direkt implementiert. Das Ziel dieser Phase ist es, sowohl Gemeinsamkeiten und Variabilität der Produktlinie sowie den Geltungssbereich und wiederverwendbare Artefakte zu identifizieren als auch die wiederverwendbaren Artefakte zu realisieren. Dazu teilt sich diese Phase in folgende Teilphasen:

1. **Product Management:** In dieser Teilphase werden zuerst der Geltungsbereich abgesteckt sowie die wesentlichen Features der SPL identifiziert und in gemeinsame und variable Features eingeteilt.

2. **Domain Requirements Engineering:** Hier werden wiederverwendbare, gemeinsame und variable Anforderungen und das Variabilitätsmodell erstellt. Das Variabilitätsmodell definiert die Variabilität der Produktlinie, indem es die Variationspunkte und Varianten der SPL und deren Abhängigkeiten definiert.

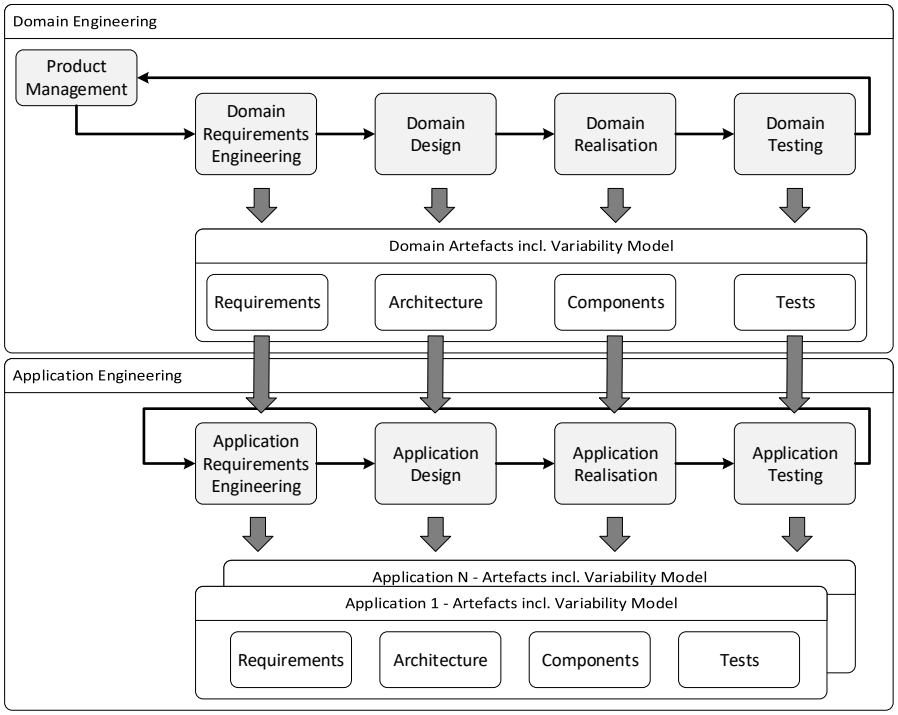

Abbildung 2.1.: Framework der Softwareproduktlinienentwicklung (nach Pohl [29])

3. **Domain Design:** In dieser Phase wird die Referenzarchitektur der Produktlinie erstellt. Die Referenzarchitektur definiert eine für alle Produkte gültige Struktur und gemeinsame Regeln zur Erstellung der Produkte.

4. **Domain Realisation:** Bei der Domain Realisation folgen der genaue Entwurf und die Implementierung von wiederverwendbaren Softwarekomponenten.

5. **Domain Testing:** Abschließend wird in dieser Teilphase untersucht, ob die Softwarekomponenten die Spezifikationen erfüllen, und es werden wiederverwendbare Testartefakte wie beispielsweise Testpläne oder Testfälle für die Komponenten entwickelt.

Application Engineering wird auch *Product Development* [9] genannt. In dieser Phase werden die Produkte erstellt, indem die im Domain Engineering implementierten Bausteine wiederverwendet werden [29]. Das Ziel ist hierbei, so viel wie möglich die bereits implementierten Bausteine wiederzuverwenden und die gegebene Variabilität auszunutzen. Es soll so wenig wie möglich zusätzlich implementiert werden, das nicht in der Produktlinie umgesetzt ist, um die Komplexität gering zu halten und eventuelle Updates nicht zu erschweren. Application Engineering unterteilt sich daher in die folgenden Teilphasen:

1. **Application Requirements Engineering:** Beim Application Requirements Engineering werden zunächst Anforderungen für das spezielle Produkt erstellt, um zu identifizieren, welche Teile des Produkts bereits durch die Produktlinie abgedeckt werden und welche eventuell darüber hinaus benötigt werden.

2. **Application Design:** In dieser Teilphase wird unter Verwendung der Referenzarchitektur die Produktarchitektur erstellt. Dabei werden die für die Produktarchitektur benötigten Teile aus der Referenzarchitektur ausgewählt und gegebenenfalls spezifisch dem Produkt angepasst.
3. **Application Realisation:** Bei der Application Realisation wird das konkrete Produkt durch Wiederverwendung der benötigten Softwarekomponenten implementiert. Auch diese müssen gegebenenfalls dem Produkt spezifisch angepasst werden.
4. **Application Testing:** Zum Schluss wird mithilfe der wiederverwendbaren und eventuell spezifisch erstellten Testartefakte das Produkt validiert und verifiziert.

Die wesentlichen Vorteile von Softwareproduktlinien entstehen durch die Wiederverwendung [29]. So ergeben sich Kostenreduzierungen und kürzere Zeiten bis zur Markteinführung, da nicht für jedes Produkt der komplette Softwareentwicklungsprozess wieder erneut durchgeführt werden muss. Des Weiteren resultiert eine höhere Qualität der Produkte, da eine bedeutend höhere Anzahl an Produkten aus demselben Entwicklungsprozess entsteht als bei Einzelentwicklung. Somit werden auch mehr Produkte getestet, bei denen Fehler gefunden werden können, und mehr Produkte verwendet, über welche dann Rückmeldungen über Probleme oder Fehler gegeben werden. Die Wiederverwendung führt dazu, dass ein gefundener Fehler in einem Produkt auch zur Verbesserung aller anderen Produkte der selben Produktlinie beiträgt und somit die Qualität der kompletten SPL und ihrer Produkte steigert.

Der grundlegende Bestandteil einer Softwareproduktlinie ist die Variabilität. Dabei unterscheidet man in einer SPL zwischen Variationspunkt und Variante [29]. *Variantionspunkte* (engl. *variation point*) sind Stellen in der SPL, an denen sich ein Feature oder ein Baustein bei unterschiedlichen Produkten unterscheiden kann. *Varianten* sind die unterschiedlichen Ausprägungsmöglichkeiten an den Variationspunkten.

Feature-Modellierung

Feature-Modelle, deren Konzept zuerst durch Kang et al. [17] vorgestellt wurde, dienen zur Erfassung von Gemeinsamkeiten und Variabilität einer Softwareproduktlinie. Sie bestehen aus hierarchisch strukturierten Features und Kompositionsrelationen zwischen diesen Features. Aus ihnen lässt sich ablesen, welche Produktvarianten - das heißt, welche möglichen Produkte - sich aus der Produktlinie ableiten lassen. Eine bestimmte Auswahl an Features aus dem Feature-Modell nennt man *Feature-Konfiguration*. Gültige Feature-Konfigurationen berücksichtigen immer die Kompositionsrelationen zwischen den Features. Die aus der gewählten Feature-Konfiguration resultierende Zusammenstellung an Grundbausteinen zur Erstellung eines Produktes wird als *Produktkonfiguration* bezeichnet. Diese repräsentiert und definiert ein eindeutiges Produkt. Eine Möglichkeit zur grafischen Darstellung von Feature-Modellen findet sich in Form von Feature-Diagrammen.

Feature-Diagramme sind in einer baumähnlichen Struktur aufgebaut und hierarchisch gegliedert, wie in Abbildung 2.2 zu sehen ist. Die Knoten des Diagramms bezeichnen Features, wobei jedes Feature durch seine Kindknoten noch weiter spezifiziert wird [4]. Kind-Features können nur ausgewählt werden, wenn auch ihre Elternknoten gewählt wurden. Features können unterschiedliche Eigenschaften annehmen, die das Ausmaß ihrer Variabilität beschreiben:

- **Mandatory-Features:** Dies sind Features, die allen Produkten der SPL gemein sind. Sie sind somit obligatorisch und müssen zwingend in jeder Feature-Konfiguration gewählt werden.

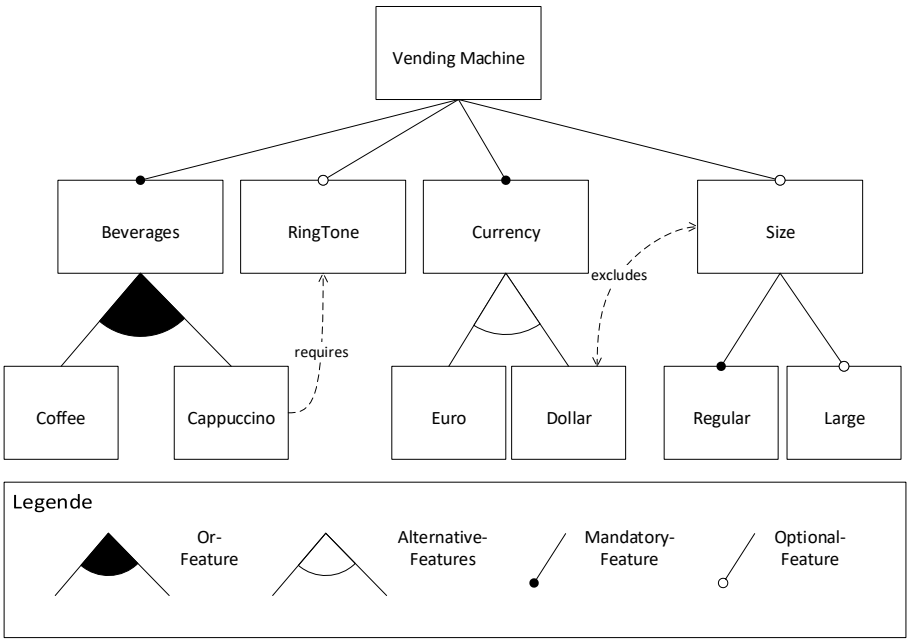

Abbildung 2.2.: Ein beispielhaftes Feature-Diagramm

- **Optional-Features:** Bei optionalen Features besteht die Möglichkeit, sie für eine Feature-Konfiguration auszuwählen oder wegzulassen. Sie bereiten somit eine Wahlmöglichkeit und sind daher in der SPL variabel.

Das Wurzel-Feature ist nicht als Mandatory-Feature gekennzeichnet, ist aber immer für jedes Produkt der Produktlinie obligatorisch. Über die genannten Eigenschaften hinaus gibt es weitere Einschränkungen bei der Auswahl von Kindknoten, die variable Stellen der SPL definieren [4]:

- **Alternative-Features:** Bei Alternative-Features darf nur genau eines der Kind-Features gewählt werden, wenn das Eltern-Feature gewählt wurde.
- **Or-Features:** Von den Or-Features können beliebig viele bis alle Features gewählt werden, mindestens eines muss jedoch gewählt werden.

Mandatory-Features werden in Feature-Diagrammen durch einen ausgefüllten Kreis oberhalb des Features dargestellt, während Optional-Features durch einen nicht ausgefüllten Kreis oberhalb des Features dargestellt werden [4]. Alternative-Features werden durch einen nicht ausgefüllten Winkel unter dem Eltern-Features dargestellt, Or-Features durch einen ausgefüllten Winkel. In der Abbildung 2.2 sind somit `Beverages`, `Currency` und `Regular` Mandatory-Features, während `RingTone`, `Size` und `Large` Optional-Features sind. `Coffee` und `Cappuccino` sind Or-Features und `Euro` und `Dollar` Alternative-Features.

Zusätzlich zu den Eltern-Kind-Beziehungen existieren außerdem ebenenübergreifende Kompositionsrelationen, welche die Beziehung zwischen zwei Features beschreiben [4]:

- **Requires-Relation:** Die Requires-Relation zeigt an, welche anderen Features ein Feature zwingend benötigt, um ordnungsgemäß funktionieren zu können.

- **Excludes-Relation:** Die Excludes-Relation definiert, welche Features unter keinen Umständen zusammen gewählt werden können, da sie sich gegenseitig ausschließen.

Die Kompositionsrelationen sind im Diagramm in Abbildung 2.2 durch gestrichelte Linien mit dem Label `exclude` beziehungsweise `requires` dargestellt. Weiterhin können diese auch durch *Constraints* in Form von aussagenlogischen Formeln dargestellt werden [4].

Eine weitere Möglichkeit Variabilität in Softwareproduktlinien grafisch darzustellen, ist das *Orthogonal Variability Model* (OVM) von Pohl et al. [29]. Das OVM beschreibt die Variabilität einer SPL, indem es die Variationspunkte der SPL und ihre Varianten definiert. Eine Variante ist mindestens einem bis beliebig vielen Variationspunkten zugeordnet und ein Variationspunkt ist ebenso mindestens einer bis beliebig vielen Varianten zugeordnet. Die Zuordnung zwischen Variationspunkt und Variante unterliegt einer *Variability Dependency*, welche die Zuordnung in zwei Arten der Abhängigkeit unterscheidet:

- **Optional Variability Dependency:** Handelt es sich bei der Zuordnung um eine Optional Variability Dependency, kann die zugeordnete Variante Teil eines Produktes sein, muss es aber nicht zwangsläufig.
- **Mandatory Variability Dependency:** Bei einer Mandatory Variability Dependency muss eine Variante immer Teil eines Produktes sein, wenn auch der zugeordnete Variationspunkt Teil des Produktes ist.

Die Optional Variability Dependency wird durch die *Alternative Choice* weiter spezifiziert [29]. Die Alternative Choice gruppiert dabei eine Menge von Varianten, die dem selben Variationspunkt optional zugeordnet sind, und definiert, wie viele Varianten mindestens gewählt werden müssen und wie viele höchstens gewählt werden dürfen. Darüber hinaus existieren in OVMs require- und exclude-Beziehungen. Diese können sowohl zwischen Varianten, zwischen Variationspunkten als auch zwischen Variante und Variationspunkt bestehen. Zusätzlich zur Möglichkeit der grafischen Darstellung der Variabilität bringt das OVM auch die Variabilitäten von unterschiedlichen Gruppen von Softwarebausteinen miteinander in Verbindung. Somit wird die Konsistenz der Variabilitätsdefinition von allen Bausteinen, wie beispielsweise Komponenten oder Anforderungen, garantiert.

Sowohl OVMs [29] als auch Feature-Modelle [4, 17] eignen sich gleichermaßen um Variabilität in Softwareproduktlinien deutlich zu machen. Im weiteren Verlauf der Arbeit werden Feature-Diagramme verwendet, um Variabilität und Gemeinsamkeiten der Fallstudien grafisch darzustellen.

Variabilitätsmodellierung von Artefakten

Die Schwierigkeit bei Softwareproduktlinien ist jedoch, die Variabilität in den Produkten ausreichend und übersichtlich in beispielsweise Komponentendiagrammen oder State Machines darzustellen. Durch die vielen verschiedenen Varianten an Produkten sowie voneinander abhängige oder sich ausschließende Features können sich sehr komplexe und unübersichtliche Modelle ergeben. Um dieses Problem zu lösen, gibt es verschiedene Ansätze Variabilität zu modellieren. Die verschiedenen, bestehenden Ansätze lassen sich in die Kategorien *annotativ*, *kompositional* und *transformational* einteilen [32]:

- **Annotativ:** Bei annotativen Ansätzen existiert ein Modell, das alle Produkte der Produktlinie in sich vereint (sogenannte 150%-Modelle). Um das Produktmodell für ein bestimmtes Produkt zu erhalten, müssen alle Teile des Modells entfernt werden, die nicht zu diesem Produkt zugehörig sind. Welche Teile des 150%-Modells zu welchen Produkten gehören, wird durch Annotationen definiert.

- **Kompositional:** Kompositionale Ansätze nutzen viele, kleine Modellfragmente, die bestimmten Features zugeordnet sind. Diese Modellfragmente werden entsprechend der in einem bestimmten Produkt verwendeten Features zu einem Produktmodell zusammengesetzt, indem nur die Fragmente für diese Features ausgewählt werden.

- **Transformational:** Transformationale Ansätze basieren auf einem Kernmodell, auf das verschiedene Regeln oder Operationen angewendet werden, die spezifizieren, wie das Kernmodell verändert werden muss, um das Produktmodell für ein bestimmtes Produkt abzuleiten.

Die verschiedenen Kategorien besitzen unterschiedliche Vor- und Nachteile [32]. Der Vorteil der annotativen Ansätze ist, dass sie eine sehr detaillierte Darstellung von Unterschieden erlauben und eine Übersicht über die verfügbaren Produktvarianten geben. Nachteil ist jedoch, dass die Modelle aufgrund fehlender Modularität sehr groß und damit wiederum unübersichtlich werden können. Hingegen haben kompositionale Ansätze durch ihren modularen Aufbau den Vorteil, dass die Modellfragmente klein und daher übersichtlich sind. Im Gegensatz dazu, sind kompositionale Ansätze teilweise unflexibel und nicht aussagekräftig genug, beispielsweise wenn Produktverhalten durch die Auswahl von bestimmten Features entfernt wird. Transformationale Ansätze vereinen die Vorteile von annotativen und kompositionalen Ansätzen ineinander, da Modularität und Flexibilität durch das Vorgehen miteinander kombiniert werden.

Aus diesem Grund wird in dieser Arbeit mit der Delta-Modellierung [7] ein transformationaler Ansatz gewählt, um sowohl Modularität als auch Flexibilität zu gewährleisten [30]. Im Folgenden soll Delta-Modellierung als Möglichkeit zur Darstellung von Variabilität in Softwareproduktlinien vorgestellt werden.

2.2. Delta-Modellierung

Delta-Modellierung [7] wird verwendet, um Variabilität in Softwareproduktlinien zu modellieren, und ist weder von Modellier- noch von Programmiersprachen abhängig [30]. Konkrete Produkte einer Softwareproduktlinie werden bei diesem Ansatz durch ein Kernmodell und eine Menge von Deltas dargestellt. Ein Delta beschreibt dabei, welche Änderungen am Kernmodell durchgeführt werden müssen, um das gewünschte Produkt zu erhalten. Diese Änderungen können das Hinzufügen, Entfernen und Modifizieren von Modellinhalten sein und werden Operationen genannt. Modellinhalte lassen sich in Modellelemente und Relationen zwischen diesen Elementen einteilen. Modellelemente können dabei beispielsweise - je nach Modellform - Komponenten, Klassen oder Zustände sein.

Das Kernmodell muss ein gültiges Produkt aus der Produktlinie sein [30]. Hierbei gibt es zu einer SPL allerdings nicht das eine richtige Kernmodell, sondern welches Produkt als Kernmodell gewählt wird, liegt im Ermessen des Modellierers. Meist handelt es sich um eine Art Standardprodukt.

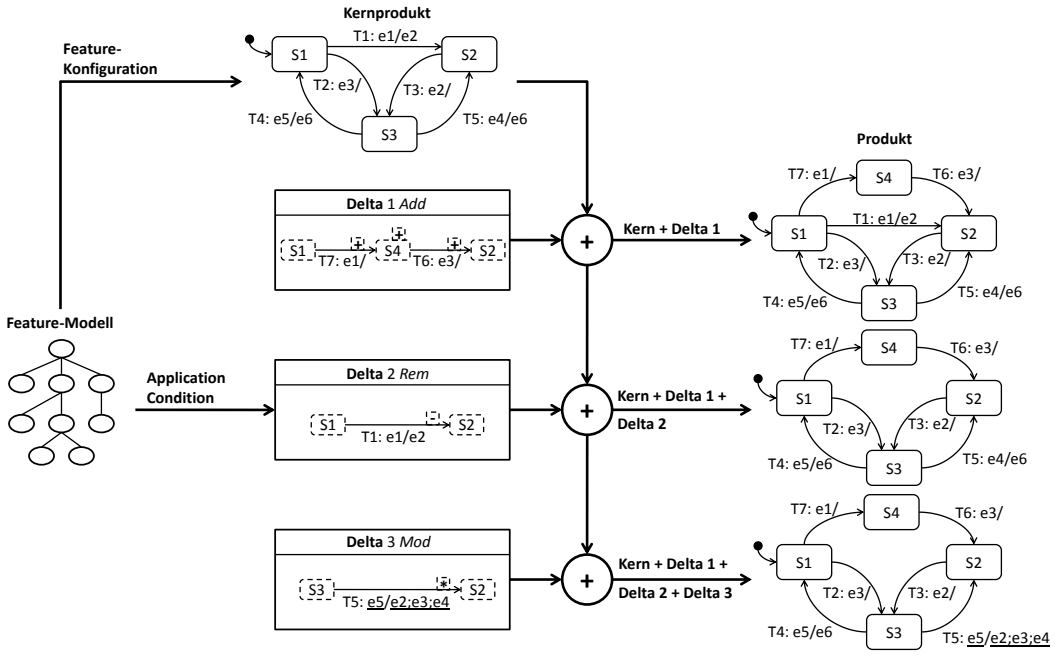

Abbildung 2.3.: Delta-Konzept am Beispiel einer State Machine [20]

Um aus dem Kernmodell ein spezifisches Produktmodell abzuleiten, werden die Operationen aus den Deltas, welche dem Produkt zugeordnet sind, auf das Kernmodell angewandt [30]. Eine beispielhafte Anwendung von Deltas auf ein Kernprodukt sieht man in Abbildung 2.3, die von Lity [20] entnommen wurde. Aus einem Feature-Modell wird eine gültige Feature-Konfiguration als Kernmodell gewählt. Auf dieses Kernmodell werden nacheinander Delta 1, Delta 2 und Delta 3 angewendet. Delta 1 fügt das Element S4 und die Transitionen T6 und T7 zum Kernmodell hinzu. Delta 2 entfernt die Transition T1 aus dem Kernmodell und Delta 3 modifiziert die Transition T5.

Die Zuordnung von Deltas zu einem bestimmten Produkt entsteht durch eine Anwendungsbedingung (engl. *application condition*), die jedem Delta angehängt ist und spezifiziert, für welche Feature-Konfiguration ein Delta angewendet werden kann [30]. Dabei bezieht sich ein Delta nicht zwangsläufig nur auf ein bestimmtes Feature, sondern kann auch einer bestimmten Kombination von mehreren Features zugeordnet sein. Es ist ebenfalls möglich, dass mehrere, unterschiedliche Deltas für dieselbe Feature-Konfiguration gültig sind. Somit müssen auch alle diese Deltas auf das Kernmodell angewendet werden, um das entsprechende Produkt mit dieser Feature-Konfiguration zu erhalten.

Bei der Anwendung von Deltas können Konflikte zwischen Änderungen entstehen [31]. So könnte in einem Delta für eine Konfiguration beispielsweise versucht werden, ein bestimmtes Element e_1 hinzuzufügen, obwohl es schon im Modell enthalten ist. Ebenso könnte versucht werden, Elemente oder Relationen zu entfernen oder zu modifizieren, die nicht im Modell vorhanden sind. Damit Konfliktfreiheit und Wohlgeformtheit von (Produkt-)Modellen in jedem Schritt garantiert werden können, muss zuvor sichergestellt werden, dass Modellinhalte, die entfernt oder modifiziert werden sollen, im Modell enthalten sind, und dass Inhalte, die hinzugefügt werden sollen, nicht enthalten sind. Dies geschieht durch eine Ordnungsrelation, die für jedes Delta definiert,

in welcher Reihenfolge es auf ein Kernmodell angewendet werden darf und welche Deltas zuvor bereits angewandt sein müssen. Alle Fallstudien in dieser Arbeit liegen delta-orientiert vor.

2.3. Evolution von Softwareproduktlinien

Während das Vorhandensein von Variabilität in einer Softwareproduktlinie als *Variabilität im Raum* bezeichnet werden kann, wird Evolution auch als *Variabilität in der Zeit* betrachtet [29]. Dies bedeutet, dass von ein und demselben Artefakt zu unterschiedlichen Zeitpunkten unterschiedliche Versionen vorliegen. Evolution beschreibt somit die Veränderungen von Artefakten über die Zeit beziehungsweise die „akkumulierten Effekte von Veränderung über die Zeit“ [23].

Evolution kann durch unterschiedliche Auslöser hervorgerufen werden [23]. Diese Auslöser können dabei erwartet sein und kontrolliert werden oder sie können unerwartet auftreten und unter Umständen unerwünscht sein. Es gibt Stellen in einer Softwareproduktlinie, an denen Evolution teilweise recht einfach umgesetzt werden kann, wie beispielsweise Variationspunkte [29]. An diesen Punkten ist Variabilität bereits vorgesehen, wodurch es vergleichsweise einfach ist, neue Varianten hinzuzufügen, Varianten zu entfernen oder zu verändern. In vielen Fällen ist es komplizierter, Evolution in einer Softwareproduktlinie umzusetzen, da häufig Abhängigkeiten zwischen Bausteinen bestehen [23]. Somit hat eine Änderung an einem Baustein auch Auswirkungen auf die abhängigen Bausteine. Dies kann eventuell zu Inkonsistenzen in der Produktlinie führen, vorallem dann, wenn sich abhängige Bausteine in widersprüchliche Richtungen entwickeln. Aus diesem Grund ist Evolution immer auch mit Risiken verbunden. Solche Risiken, die das Scheitern einer kompletten Produktlinie zur Folge haben können, sind das Stören von Konsistenz, Vollständigkeit und Korrektheit der SPL.

- **Konsistenz:** Die Konsistenz einer Produktlinie ist gefährdet, wenn sich in Beziehung zueinander stehende Bausteine in unterschiedliche Richtungen entwickeln. Dies kann dazu führen, dass die Bausteine nicht länger miteinander kompatibel sind und ein fehlerfreies Zusammenarbeiten nicht mehr möglich ist.

- **Vollständigkeit:** Wird eine große Anzahl an Bausteinen im Evolutionsprozess verändert, kann es passieren, dass Verbindungen zwischen diesen Bausteinen versehentlich verloren gehen. Dies kann zur Folge haben, dass Bausteine, die durch ebenjene Verbindungen einer bestimmten Konfiguration zugeordnet waren, nicht mehr in dieser Konfiguration auftauchen. Auf diese Weise würden diese Bausteine auch nicht mehr für Veränderungen in Betracht gezogen werden, die durch die zuvor verbundenen Bausteine propagiert werden müssten. Darüber hinaus besteht die Möglichkeit, dass ungewollt komplette Bausteine aus der Produktlinie entfernt werden.

- **Korrektheit:** Es besteht die Möglichkeit, dass eine Veränderung an einem Baustein zu einem Defekt des Bausteins führt, statt ihn zu optimieren.

Die Wahrscheinlichkeit der genannten Risiken erhöht sich, je mehr Bausteine die Produktlinie enthält und je komplexer die Beziehungen zwischen den Bausteinen sind [23]. Aus diesem Grund ist es nötig, bereits vor der Umsetzung einer Änderung zu wissen, welche Auswirkungen diese auf die Produktlinie haben wird, und die Auswirkungen im Evolutionsplan festzuhalten. So kann be-

spielsweise sichergestellt werden, dass das Entfernen eines Bausteins aus der Produktlinie geplant und kein Versehen war.

Die Auswirkungen, die eine Änderung auf die SPL hat, werden durch die *Change Impact Analyse* identifiziert [23]. Zuerst wird hierbei der Baustein bestimmt, an dem die erste Änderung zu Tage tritt. Dann werden alle Beziehungen betrachtet, die dieser Baustein zu anderen Bausteinen hat, und die in Beziehung stehenden Bausteine daraufhin untersucht, ob die Änderungen ebenfalls Auswirkungen auf sie haben. Sollte eine Änderung Auswirkungen auf andere Bausteine haben, wird geprüft welche daraus resultierenden Veränderungen an diesen Bausteinen vorgenommen werden müssen. Des Weiteren muss auch für diese Bausteine wieder identifiziert werden, welche anderen Bausteine mit ihnen in Verbindung stehen und ob sich auch dort wiederum Änderungen ergeben. Auf diese Weise wird bereits vor der Umsetzung einer gewünschten Änderung der Effekt bestimmt, den die Änderung auf die komplette Produktlinie hat. So lässt sich darauf schließen, ob Aufwand und Nutzen in einem positiven Verhältnis zueinander stehen.

Zusätzlich zur Untersuchung der möglichen Auswirkungen einer zukünftigen Evolution auf die SPL ist auch die Betrachtung der Evolutionshistorie von Interesse [5]. Diese ermöglicht eine Dokumentation der bereits erfolgten Änderungen und erlaubt eine Analyse der Historie und aufgrund dessen gezieltere Vermutungen für die weitere Entwicklungsrichtung der Produktlinie. Auch während der Implementierung von Evolution müssen die umzusetzenden Änderungen ausführlich beschrieben sein, um Fehler in der Umsetzung zu vermeiden. Aus den genannten Gründen ist es notwendig, Veränderungen durch Evolution eindeutig spezifizieren sowie festhalten und dementsprechend modellieren zu können.

Evolution kann auf unterschiedlichen Abstraktionsebenen, wie etwa auf der Ebene von Artefakten oder auch der SPL als Ganzem, beschrieben werden [5]. Die generelle Beschreibung der Änderungen zwischen zwei unterschiedlichen Versionen eines Artefakts lässt sich in zwei Kategorien einteilen:

- **Difference Models:** Difference Models modellieren die Unterschiede zwischen zwei Versionen eines Artefakts, indem die evolutionären Änderungen durch das Hinzufügen, Entfernen oder Modifizieren von Modellinhalten beschrieben werden.
- **Change Operators:** Durch Change Operators werden Operationen beschrieben, die an einem Modell vorgenommen werden müssen, um eine bestimmte Änderung zu erzielen. Wie bei Difference Models existieren auch hier Operatoren für Hinzufügen, Entfernen und Modifizieren, jedoch lassen sich Änderungen auch durch andere, komplexere Operatoren beschreiben.

Des Weiteren gibt es Ansätze, welche diese beiden Konzepte miteinander kombinieren [5]. Neben dem Ansatz von Acher et al. [1] zur Identifizierung von semantischen und syntaktischen Unterschieden zwischen Feature-Modellen und den *Change Sets* von Hendrickson et al. [15] gehört auch Delta-Modellierung [7] (vgl. Kapitel 2.2) in die Kategorie der Difference Models. Die Change Sets genannten Difference Models von Hendrickson et al. [15] bieten die Möglichkeit die Architekturen von verschiedenen Produkten durch die Kombination von Change Sets zu spezifizieren. In der Kategorie Change Operators stellen beispielweise Alves et al. [2] Change Operators für Feature-Modelle vor, die ein Refactoring von SPLs erlauben, ohne semantische Änderungen vorzunehmen. Auch wenn diese Ansätze nicht ausdrücklich auf Evolution ausgerichtet sind, sondern eine generelle Beschreibung der Variabilität in Produktlinien zum Ziel haben, lässt sich doch erkennen, dass sich ähnliche

Mechanismen ebenfalls für die Variabilität in der Zeit anwenden lassen können. Ein kombinierter Ansatz beider Kategorien speziell für Evolution findet sich beim EvoFM von Botterweck et al. [6].

Durch EvoFM [6] werden Feature-Modelle so erweitert, dass sie ebenfalls ihre Entwicklung über die Zeit festhalten. Dazu werden sämtliche mögliche Evolutionsoperationen definiert, um Rückschlüsse auf die Auswirkungen der Evolution zu vereinfachen. Auch temporale Feature-Modelle von Nieke et al. [27] erweitern normale Feature-Modelle, sodass sie Evolution erfassen, und bieten eine entsprechende Change Impact Analyse zur Überprüfung der Validität von Feature-Kofigurationen. Sowohl EvoFM [6] als auch temporale Feature-Modelle [27] lassen sich allerdings nicht auf andere Typen von Softwareartefakten übertragen, sondern funktionieren lediglich für Feature-Modelle.

Für annotative und kompositionale Softwareproduktlinien stellen Neves et al. [26] für eine sichere Evolution Templates vor, welche sicherstellen, dass durch die Anwendung von Evolutionsoperationen keine existierenden Varianten auf eine unerwünschte Weise beeinflusst werden. Allerdings bieten die Templates keine gemeinsame Methode zur Behandlung von Variabilität in Raum und Zeit. Auch eine Evolutionshistorie existiert lediglich indirekt und muss basierend auf den Mengen von angewendeten Templates abgeleitet werden.

Da in dieser Arbeit Delta-Modellierung zur Darstellung von Variabilität in Softwareproduktlinien verwendet wird, wird zur Modellierung von Evolution in SPLs ein Ansatz gewählt, der auf dem Konzept der Delta-Modellierung basiert. Verschiedene Ansätze, die Delta-Modellierung als Grundlage nutzen, sind etwa Higher-Order Delta-Modellierung von Lity et al. [21] oder die Ansätze von Haber et al. [14], Lima et al. [19], Dhungana et al. [10] und Seidl et al. [33]. Higher-Order Delta-Modellierung [21] wendet das Konzept der Delta-Modellierung auf komplette Delta-Modelle statt einfacher Kernmodelle an. Eine ausführliche Beschreibung von Higher-Order Delta-Modellierung ist in Kapitel 2.4 zu finden. Haber et al. [14] stellen einen Ansatz zur Modellierung von Variabilität in Zeit und Raum mit Fokus auf SPL-Architekturen vor, bei dem ebenfalls Änderungen an Delta-Mengen vorgenommen werden. Dieser Ansatz bietet zusätzlich eine Refactoring-Möglichkeit, um eine Degenerierung der SPL zu verhindern, hält jedoch nicht die Historie der Evolutionsentwicklung fest. Beim Ansatz von Lima et al. [19] werden Code-Bausteinen bestimmte Features zugeordnet und mögliche Konflikte, die durch eine gegensätzliche Entwicklung von diesen entstehen, werden durch die Erstellung eines Delta-Modells gelöst, das die Konflikte identifiziert. Hierbei wird allerdings immer nur ein Evolutionsschritt dargestellt. Dhungana et al. [10] nutzen - wie auch Lima et al. [19] - Deltas, um die Unterschiede zwischen verschiedenen SPL-Versionen zu erfassen und mögliche Konflikte aufzuspüren und aufzulösen. Beide Ansätze ermöglichen Change Impact Analysen, offerieren jedoch keine umfassende Methode, um gleichermaßen mit räumlicher und zeitlicher Variabilität umzugehen. Seidl et al. [33] bieten einen Formalismus, welcher - im Gegensatz zu Higher-Order Delta Modellierung - sowohl Deltas zur Darstellung von Variabilität als auch Deltas zur Umsetzung von Evolution auf der selben Modellierungsebene anwendet. Dieser Ansatz ist zwar generisch und kann damit für unterschiedliche Artefakttypen angewendet werden, gestattet allerdings keine Change Impact Analyse. Für detaillierte Informationen zu den unterschiedlichen Ansätzen wird auf die genannte Literatur verwiesen [1, 2, 5, 6, 10, 14, 15, 19, 26, 27, 33].

Aufgrund der Tatsache, dass die verwendeten Fallstudien bereits delta-orientiert vorliegen, und der Möglichkeit, eine komplette Evolutionshistorie darzustellen, wird in dieser Arbeit Higher-Order Delta-Modellierung verwendet, welche im folgenden Kapitel genauer vorgestellt wird.

2.4. Higher-Order Delta-Modellierung

Higher-Order Delta-Modellierung [21] ist ein Ansatz Evolution in Softwareproduktlinien zu modellieren. Dieser basiert auf Delta-Modellierung [7] und hebt deren Konzept auf die nächsthöhere Ebene, um komplette Delta-Modelle und nicht nur Kernmodelle zu verändern.

Higher-Order Deltas (HODs) setzen sich aus Evolutionsoperationen zusammen [21]. Diese Operationen sind, wie bei der klassischen Delta-Modellierung auch, das Hinzufügen, das Entfernen und das Modifizieren. Allerdings werden durch Higher-Order Deltas nicht wie bei normalen Deltas einzelne Modellelemente hinzugefügt, entfernt oder modifiziert, sondern komplette Deltas. Auf diese Weise wird das gesamte Delta-Modell verändert und somit auch die Menge an Produktmodellen, die sich daraus ableiten lässt.

HODs fügen also entweder gänzlich neue Deltas zum Delta-Modell hinzu, entfernen komplette Deltas, die nicht mehr benötigt werden, aus dem Delta-Modell oder modifizieren bestehende Deltas [21]. Bei der Modifizierung eines Deltas wird dessen beinhaltete Menge an Operationen verändert. Das heißt, es können im Delta enthaltene Operationen entfernt werden oder neue Operationen hinzugefügt werden.

Evolution von Delta-Modellen ergibt sich beispielsweise dadurch, dass Anforderungen geändert oder Features zur Produktlinie hinzugefügt oder entfernt werden [21]. Wird ein Feature entfernt, so könnten etwa Deltas wegfallen, die mit diesem Feature in Zusammenhang standen und daher nicht mehr benötigt werden. Kommt ein neues Feature hinzu, könnten neue Deltas ergänzt werden, um neue Funktionen umsetzen zu können, oder Deltas modifiziert werden, um sie an die neuen Funktionen anzupassen. Auch bei der Veränderung von bestehenden Features, wenn diese beispielsweise in ihrer Funktionalität erweitert oder beschränkt werden, kann ein Hinzufügen, Entfernen oder Modifizieren von Deltas nötig sein.

Dadurch verändert sich automatisch auch die Menge an Produktmodellen, die sich durch die unterschiedlichen Kombinationen von Kernmodell und Deltas ergibt [21]. Werden Deltas aus dem Delta-Modell entfernt, fallen auch alle Produktmodelle weg, die durch Anwendung dieser Deltas auf das Kernmodell entstanden sind. Ebenso kommen neue Produktmodelle hinzu, wenn neue Deltas zum Delta-Modell hinzugefügt werden. Auch durch das Modifizieren von Deltas können neue Produktmodelle hinzugefügt und bestehende Produktmodelle modifiziert oder entfernt werden.

Jede Entwicklung eines Delta-Modells in eine neue Version lässt sich als ein Evolutionsschritt in der Evolutionshistorie eines Delta-Modells betrachten [21]. Higher-Order Deltas spezifizieren demnach die zeitliche Variabilität, und somit die Evolution, von kompletten Delta-Modellen, indem sie die Veränderungen zwischen zwei zeitlich aufeinanderfolgenden Delta-Modellen beschreiben. Evolutionsschritte und komplette Evolutionshistorien werden dabei in Form von zwei unterschiedlichen Typen von Delta-Modellen festgehalten. *Evolution-Step Higher-Order Delta-Modelle* setzen sich zusammen aus einem Delta-Modell und genau einem Higher-Order Delta, welches die Veränderungen für diesen Evolutionsschritt beschreibt. *Evolution-History Higher-Order Delta-Modelle* bestehen aus einem Delta-Modell und einer Sequenz von Higher-Order Deltas. Diese Sequenz von HODs wird inkrementell auf das Delta-Modell angewendet. Auf diese Weise wird die komplette Evolutionshistorie dargestellt, wobei jeder Schritt einzeln betrachtet werden kann, indem die Higher-Order Deltas nacheinander auf das jeweils durch ihr Vorgänger-HOD erzeugte Delta-Modell angewendet werden.

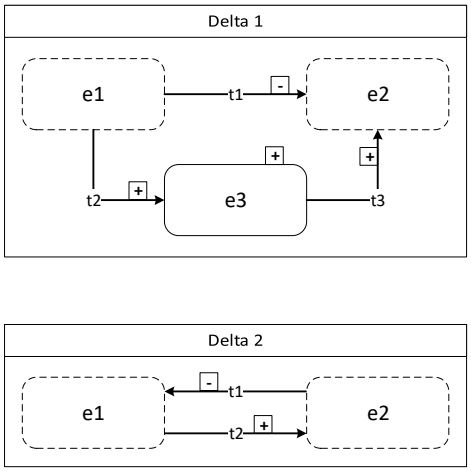

Abbildung 2.4.: Delta-Menge

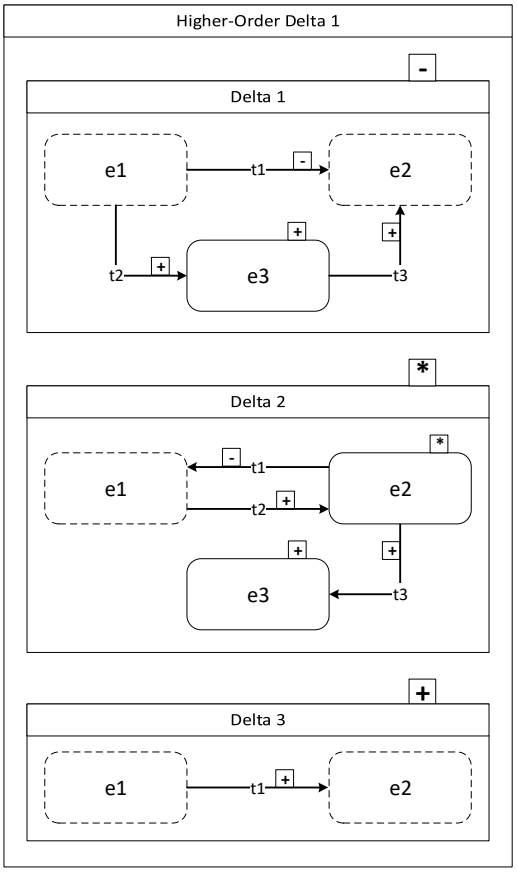

Abbildung 2.5.: Higher-Order Delta

Die Abbildungen 2.4 und 2.5 zeigen die beispielhafte Anwendung von Higher-Order Deltas. Abbildung 2.4 stellt eine Menge von Deltas dar, die aus den Deltas `Delta 1` und `Delta 2` besteht. `Delta 1` fügt das Element `e1` und die Transitionen `t2` und `t3` hinzu und entfernt die Transition `t1`. `Delta 2` entfernt die Transition `t1` und fügt die Transition `t2` hinzu.

Das HOD `Higher-Order Delta 1` in Abbildung 2.5 entfernt `Delta 1` aus der Menge von Deltas. Zugleich modifiziert es `Delta 2`, sodass dieses nicht nur `t1` entfernt und `t2` hinzufügt, sondern auch noch Element `e1` und Transition `t3` ergänzt sowie Element `e2` modifiziert. Des Weiteren fügt das HOD das neue Delta `Delta 3` zur Delta-Menge hinzu. Die Menge von Deltas besteht somit nach Anwendung von `Higher-Order Delta 1` nur noch aus `Delta 3` und dem modifizierten `Delta 2`.

3 Fallstudien

In diesem Kapitel werden zunächst die vier verschiedenen, in dieser Arbeit verwendeten Fallstudien mit ursprünglichem Verhalten vorgestellt. Des Weiteren wird zu jeder Fallstudie ein Delta-Modell beschrieben, das für die Entwicklung von Evolutionsszenarien benötigt wird. Die Delta-Modelle bestehen aus einem Kern und verschiedenen Deltas, die, zu Sequenzen zusammengefügt, gültige Produktvarianten ergeben. Die Fallstudien sind ein Body Comfort System [22], ein Verkaufsautomat, eine Scheibenwischanlage und eine Minenpumpanlage [8].

3.1. Verkaufsautomat

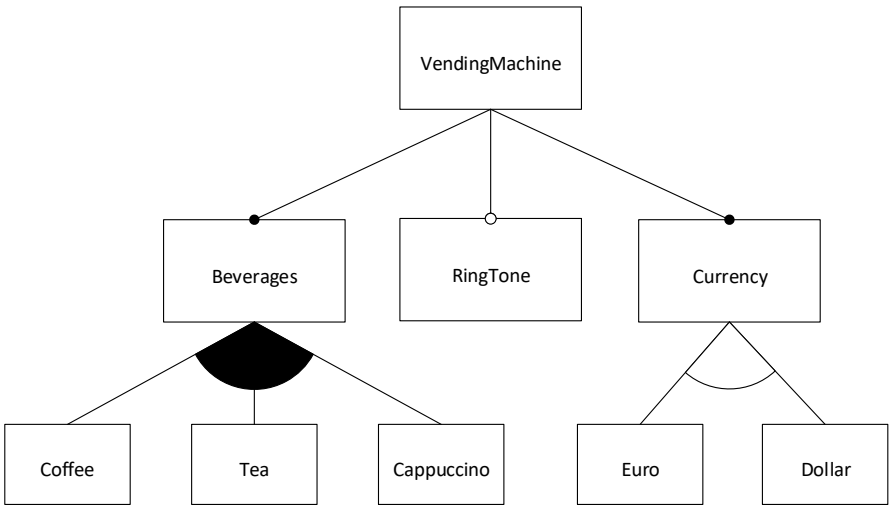

Abbildung 3.1.: Feature-Diagramm des Verkaufsautomaten (nach [8])

Der Verkaufsautomat [8] beschreibt einen Automaten, der Heißgetränke verkauft. Er wurde zuerst von Fantechi und Gnesi [11] mit zwei unterschiedlichen Produktvarianten beschrieben und von Classen [8] erweitert. Die Produktlinie des Verkaufsautomaten bietet verschiedene Getränke (Kaffee, Tee und Cappuccino) an und akzeptiert zwei unterschiedliche Währungen (Euro und Dollar). Zu sehen ist dieser Zusammenhang im Feature-Modell in Abbildung 3.1. Für ein gültiges Produkt muss mindestens ein Getränk ausgewählt werden und die Getränke können beliebig miteinander kombiniert werden. Des Weiteren darf nur genau eine Währung ausgewählt werden. Darüber hinaus existiert das optionale Feature `Ring Tone`, durch das ein Ton erklingt, wenn das Getränk fertig ausgegeben wurde.

Verhalten

Der Verkaufsautomat [8] wird immer durch den Einwurf einer Münze aktiviert. Danach muss entschieden werden, ob Zucker gewünscht ist oder nicht. Nach dieser Entscheidung wird das favo-

risierte Getränk ausgewählt. Wurde zuvor Zucker ausgewählt, wird nun zuerst der Zucker in den Becher gefüllt und dann das gewählte Getränk. Wurde kein Zucker gewählt, wird direkt das Getränk eingefüllt. Danach erscheint auf dem Display eine Nachricht, dass das Getränk fertiggestellt wurde und es wird ein Ton ausgegeben. Falls ein Automat das Feature `Ring Tone` nicht besitzt, wird die Tonausgabe übersprungen. Nachdem der Becher aus dem Automaten genommen wurde, kehrt der Automat in seinen Ausgangszustand zurück und wartet auf den nächsten Münzeinwurf.

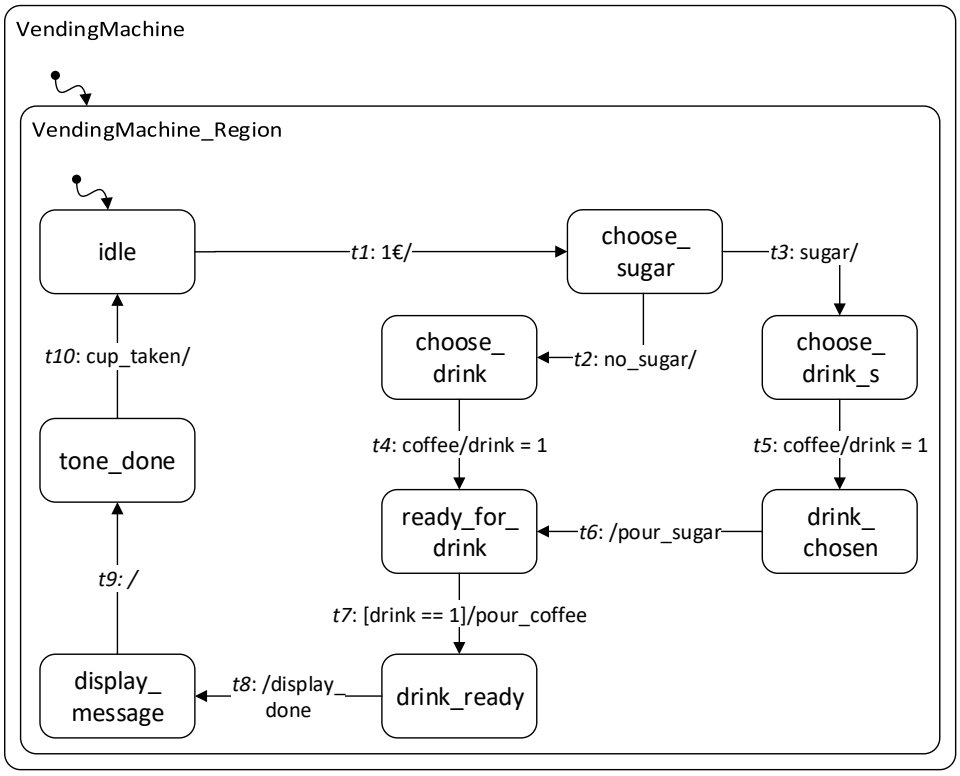

Abbildung 3.2.: Kernmodell des Verkaufsautomaten

Als Kernmodell für den Verkaufsautomaten wird in dieser Arbeit ein Automat gewählt, der Euro als Währung akzeptiert, Kaffee als Getränk anbietet und keinen Ton ausgibt. Der Kern ist in Abbildung 3.2 zu sehen und zeigt das zuvor beschriebene Verhalten für die gewählten Features `Coffee` und `Euro` anhand einer State Machine.

In Abbildung 3.3 sind die Deltas zu sehen, die benötigt werden, um aus dem Kernmodell die Produktmodelle für andere gültige Verkaufsautomaten abzuleiten. Um einen Automaten zu erzeugen, der einen Ton ausgibt, wenn das Getränk ausgegeben wurde, wird `Delta_Ringtone` verwendet. Dieses Delta ersetzt die Transition `t9` zum Überspringen der Tonausgabe durch die Transition `t11`, welche die Tonausgabe auslöst. Falls Dollar statt Euro als Währung gewünscht ist, entfernt `Delta_Dollar` die Transition `t1` für Euro zwischen den Zuständen `idle` und `wait_for_sugar` und fügt stattdessen die Transition `t12` für Dollar hinzu.

`Delta_Tea` beschreibt, welche Änderungen am Kernmodell vorgenommen werden müssen, um einen Automaten zu erhalten, der zusätzlich Tee als Getränk anbietet. Hierzu werden die Transitionen `t13` und `t14` für die Teeauswahl sowie die Transition `t15` zum Einfüllen von Tee eingefügt.

Sollte Kaffee nicht als Getränk zur Auswahl gewünscht sein, kann er durch `Delta_Remove_Coffee` entfernt werden. Dazu werden die Transitionen `t4`, `t5` und `t7` entfernt. `Delta_Cappuccino` fügt Cappuccino als Getränk zur Auswahl dazu, indem der Zustand `ca_ready_for_milk` sowie die Transitionen `t17`, `t18`, `t19` und `t20` zum Auswählen von Cappuccino und Einfüllen von Kaffee und Milch eingefügt werden.

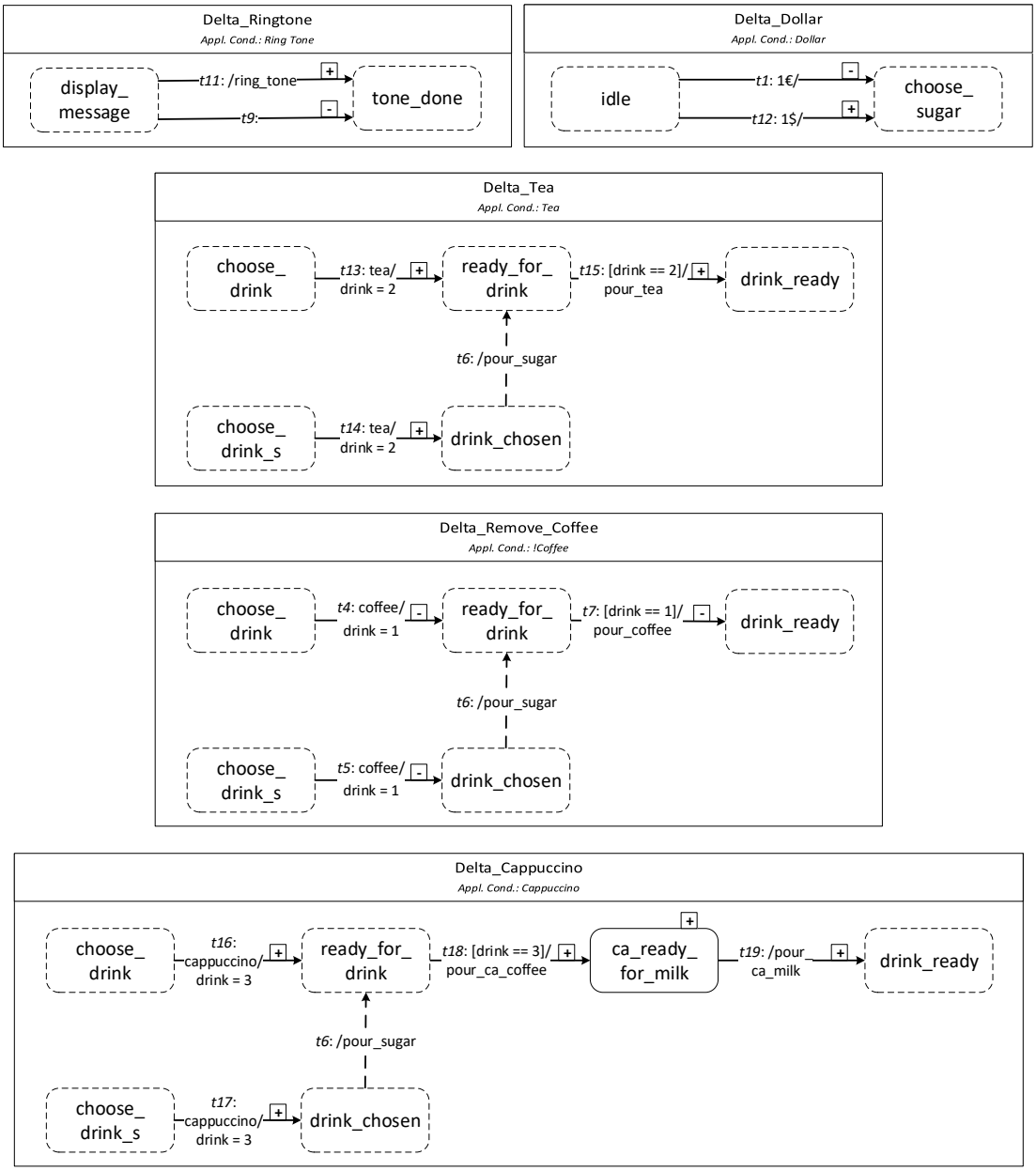

Abbildung 3.3.: Deltas des Verkaufsautomaten

Der Veraufsautomat besitzt insgesamt 28 gültige Feature-Konfigurationen und dementsprechend auch 28 verschiedene Produktvarianten.

3.2. Scheibenwischanlage

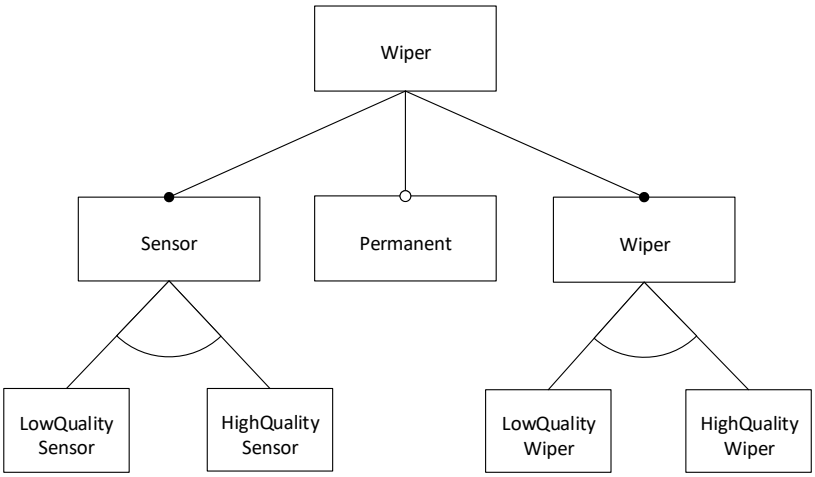

Abbildung 3.4.: Feature-Diagramm der Scheibenwischanlage (nach [8])

Die Scheibenwischanlage [8] beschreibt einen einfachen Autoscheibenwischer mit Regensensor. Erstmals wurde die Scheibenwischanlage von Gruler et al. [12] beschrieben und später von Classen [8] angepasst. Wie Abbildung 3.4 zeigt, besteht die Scheibenwischanlage aus einem Sensor und einem Wischer. Der Sensor sowie der Wischer sind in den zwei Qualitätsstufen *High* und *Low* erhältlich. Zusätzlich gibt es noch das optionale Feature `Permanent`, welches ein durchgängiges Wischen erlaubt.

Verhalten

Der minderwertige Sensor erkennt lediglich den Unterschied zwischen Regen oder keinem Regen. Der minderwertige Wischer reagiert auf Regen automatisch mit einem langsamen Wischintervall. Der hochwertige Sensor erkennt sowohl keinen Regen als auch einen Unterschied zwischen leichtem Regen und starkem Regen. Der hochwertige Wischer aktiviert dementsprechend automatisch ein langsames oder ein schnelles Wischintervall. Besitzt die Scheibenwischanlage das optionale Feature `Permanent`, kann zusätzlich statt dem automatischen Wischen manuell ein permanentes Wischen aktiviert werden.

Das Kernmodell für die Scheibenwischanlage besteht aus dem minderwertigen Wischer und dem minderwertigen Sensor. Der Kern ist in Abbildung 3.5 abgebildet und besteht aus den zwei parallel laufenden State Machines `Wiper_Root` und `Sensor_Root`, die jeweils das Verhalten von Wischer beziehungsweise Sensor abbilden. Erkennt der Sensor keinen Regen, geht er in den Zustand `sense_no_rain` und sendet die Nachricht `noRain` an den Wischer. Sowohl bei leichtem als auch bei starkem Regen sendet der Sensor die Nachricht `rain` und wechselt in den Zustand `sense_rain`. Wird der Wischer eingeschaltet, wechselt er in den Zustand `enabled`. Dort löst er bei der Nachricht `rain` ein langsames Wischen aus und stoppt dieses wiederum bei der Nachricht `noRain`. Im Zustand `disabled` ist der Wischer ausgeschaltet und reagiert nicht auf Nachrichten.

Abbildung 3.6 zeigt die Deltas, mit denen sich weitere gültige Produktmodelle aus dem Kernmodell ableiten lassen.

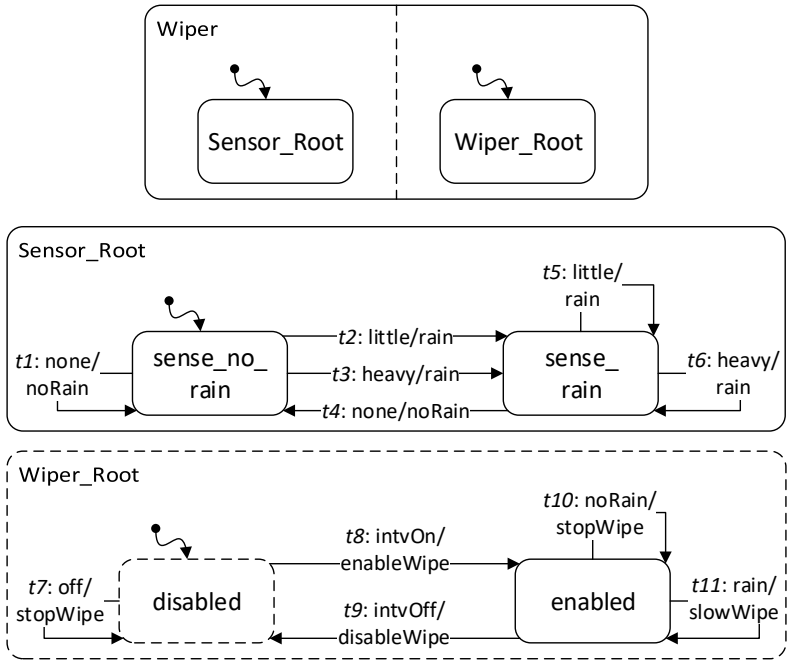

Abbildung 3.5.: Kernmodell der Scheibenwischanlage

`Delta_HighSensor` nimmt die notwendigen Änderungen vor, um aus dem minderwertigen Sensor einen hochwertigen Sensor abzuleiten. Dazu wird der Zustand `sense_heavy_rain` hinzugefügt, in welchen der Sensor durch die neuen Transitionen `t13` oder `t15` wechselt, wenn er starken Regen erkennt. Durch die neue Transition `t14` verbleibt er bei starkem Regen im genannten Zustand. Alle drei Transitionen lösen die Nachricht `heavyRain` aus. Des Weiteren wechselt er bei keinem Regen durch die hinzugefügte Transition `t12` mit der Nachricht `noRain` in den Zustand `sense_no_rain` und durch `t15` mit der Nachricht `rain` in `sense_rain`. Darüber hinaus werden die überflüssigen Transitionen `t3` und `t6` entfernt, die sonst bei starkem Regen weiterhin die Nachricht `rain` senden würden.

`Delta_HighSensor_LowWiper` passt den minderwertigen Wischer an den hochwertigen Sensor an, indem es die Transition `t17` einfügt. Durch diese reagiert der Wischer in eingeschaltetem Zustand auf die Nachricht `heavyRain` mit einem langsamen Wischen. Dieses Delta kann nur nach `Delta_HighSensor` angewendet werden.

`Delta_HighSensor_HighWiper` beschreibt die notwendige Änderung, um aus dem minderwertigen Wischer einen hochwertigen Wischer zu machen, wenn zuvor bereits durch `Delta_HighSensor` aus dem minderwertigen Sensor ein hochwertiger Sensor abgeleitet wurde. Hierzu wird die Transition `t18` hinzugefügt, die den eingeschalteten Wischer auf die Nachricht `heavyRain` mit einem schnellen Wischen reagieren lässt.

Ist die Kombination aus minderwertigem Sensor und hochwertigem Wischer gewünscht, findet `Delta_LowSensor_HighWiper` Anwendung. Dieses Delta entfernt die Transition `t11` und ersetzt sie durch die Transition `t19`. Diese lässt den Wischer mit einem schnellen statt mit einem langsamen Wischen auf die Nachricht `rain` reagieren.

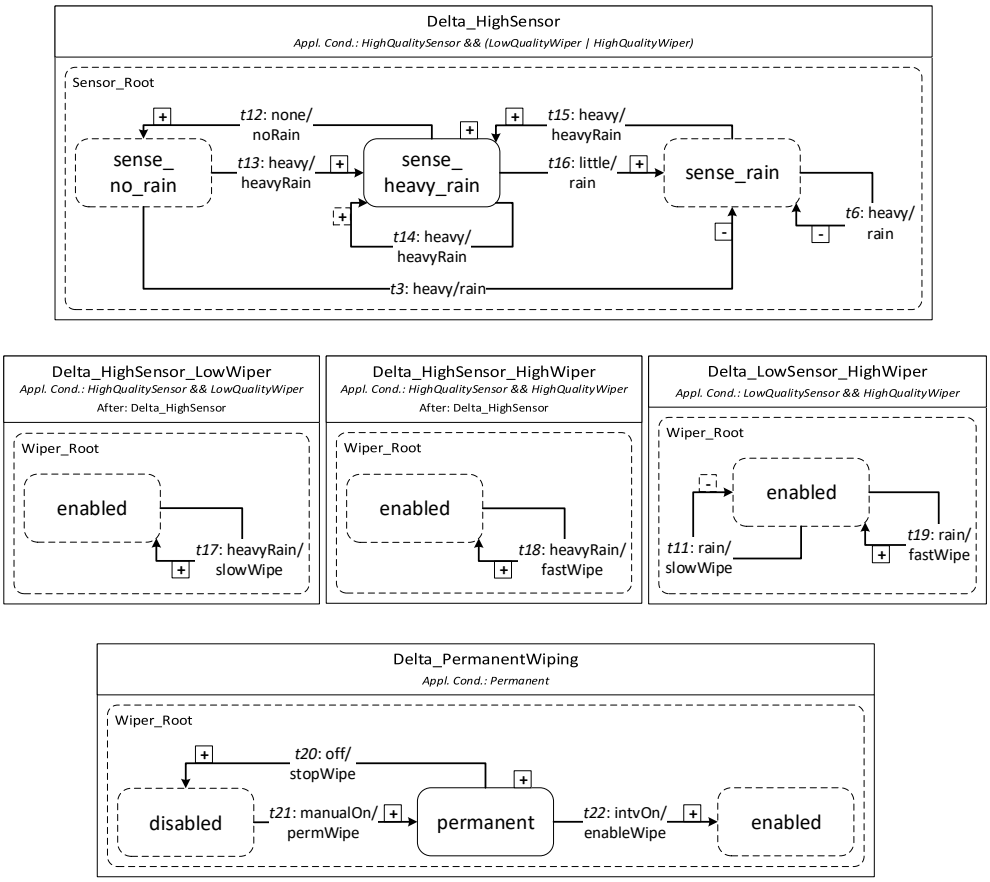

Abbildung 3.6.: Deltas der Scheibenwischanlage

Zu allen vier Kombinationen aus Sensor und Wischer kann durch `Delta_PermanentWiping` eine permanente Wischfunktion hinzugefügt werden. Dazu wird in `Wiper_Root` der Zustand `permanent` hinzugefügt. Ist das permanente Wischen aktiviert, kann der Wischer durch die neue Transition `t20` komplett ausgeschaltet werden, und durch die neue Transition `t21` kann bei ausgeschaltetem Wischer das permanente Wischen wieder eingeschaltet werden. Darüber hinaus lässt sich durch die hinzugefügte Transition `t22` von permanentem Wischen auf automatisches Intervall umschalten.

Die Scheibenwischanlage besitzt im Ganzen acht mögliche Feature-Konfigurationen und somit acht unterschiedliche Produktvarianten.

3.3. Minenpumpanlage

Die Minenpumpanlage [8] dient dazu, Wasser aus einem Minenschacht zu pumpen. Sie wurde bereits 1983 von Kramer et al. [18] vorgestellt und von Classen [8] erweitert. Die Produktlinie der Minenpumpanlage unterstützt das Mandatory-Feature `Water Regulation` und die Optional-Features `Command` und `Methane Detection`, abgebildet im Feature-Diagramm in Abbildung 3.7. Das Feature `Water Regulation` liefert einen Sensor zur Überwachung des Wasserstandes und eine Pumpe zum Abpumpen des Wassers. Dabei kann durch die Kind-Features `Low`, `Normal` und `High` die Behandlung des Wasserstandes variiert werden. Die Behandlung von hohem Wasserstand durch `High` ist zwingend erforderlich, während die Behandlung von normalen oder niedrigen Wasserständen durch

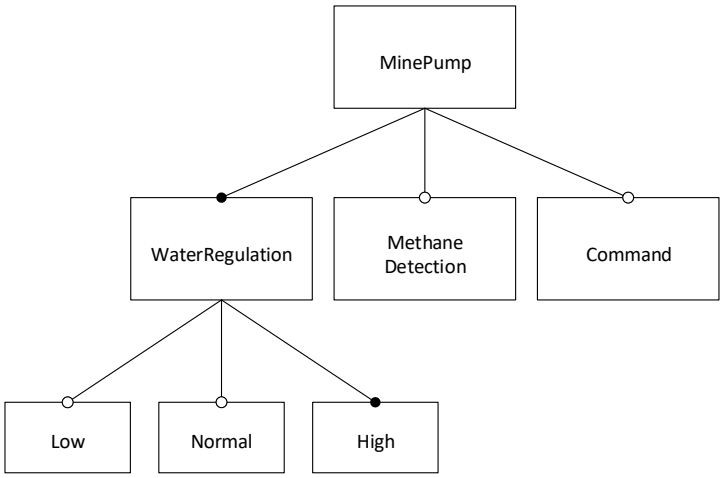

Abbildung 3.7.: Feature-Diagramm der Minenpumpanlage (nach [8])

`Normal` und `Low` optional wählbar ist. Das Feature `Command` bietet die Möglichkeit, die Funktion von `Water Regulation` ein- und auszuschalten. `Methane Detection` stellt ein Alarminterface und einen Methansensor, um eine Explosion von Methangas durch Einschalten der Pumpe zu verhindern.

Verhalten

Die Minenpumpanlage [8] überwacht mit dem Sensor durchgehend den Wasserstand. Bei hohem Wasserstand wird die Pumpe eingeschaltet und nach dem Ablauf einer bestimmten Zeitspanne wieder ausgeschaltet. Wurde das Feature `Low` gewählt, schaltet sich die Pumpe außerdem automatisch ab, wenn der Wasserstand niedrig ist. Das Feature `Normal` führt bei normalem Wasserstand keine Aktion aus. Bei gewählter `Methane Detection` wird außerdem ständig das Methanlevel überwacht. Wird in diesem Fall der Wasserstand als hoch erkannt, wird nun vor dem Einschalten der Pumpe das Methanlevel überprüft. Bei Überschreiten eines bestimmten Methanschwellenwertes wird die Pumpe trotz hohem Wasserstand nicht eingeschaltet, um eine mögliche Explosion zu vermeiden. Weiterhin wird die Pumpe ausgeschaltet, sollte sie bereits laufen, wenn der zulässige Methanwert überschritten wird. Wird die Pumpe über `Stop` ausgeschaltet, führt sie keine Überwachungs- und Pumpfunktionen mehr aus, bis sie wieder eingeschaltet wird.

Das Kernmodell für die Minenpumpanlage ist in Abbildung 3.8 dargestellt. Es umfasst die State Machine für die Funktionalität zur Überwachung und Behandlung von hohem Wasserstand. Dazu teilt sich die State Machine in drei parallel laufende Substate Machines `Pump_Ctrl_Region`, `Water_Monitoring_Region` und `MinePump_System_Region`. `Pump_Ctrl_Region` schaltet die Pumpe bei Nachricht über entsprechenden Wasserstand ein beziehungsweise aus und `Water_Monitoring_Region` überwacht den Wasserstand und sendet Nachrichten mit der Information über niedrigen, normalen oder hohen Wasserstand. `MinePump_System_Region` wertet in `Water_Level_Control_Region` die Nachrichten über den Wasserstand aus. Bei hohem Wasserstand wird geprüft, ob die Pumpe bereits eingeschaltet ist, und falls nicht, wird die Nachricht gesendet, dass die Pumpe eingeschaltet werden soll. Zusätzlich wird hier geprüft, wie lange die eingeschaltete Pumpe schon läuft, und bei einer gewissen Zeitüberschreitung ausgeschaltet.

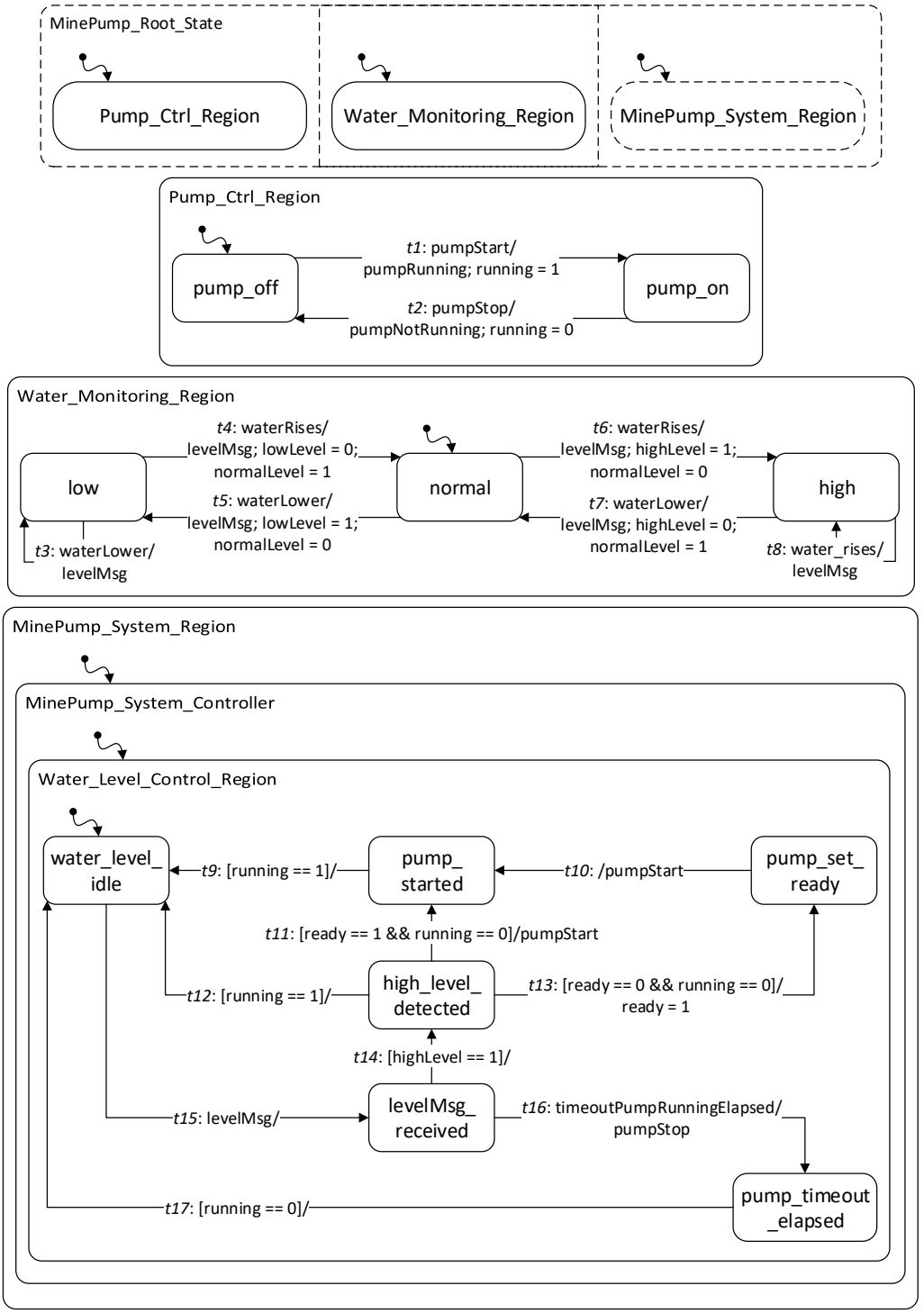

Abbildung 3.8.: Kernmodell der Minenpumpanlage

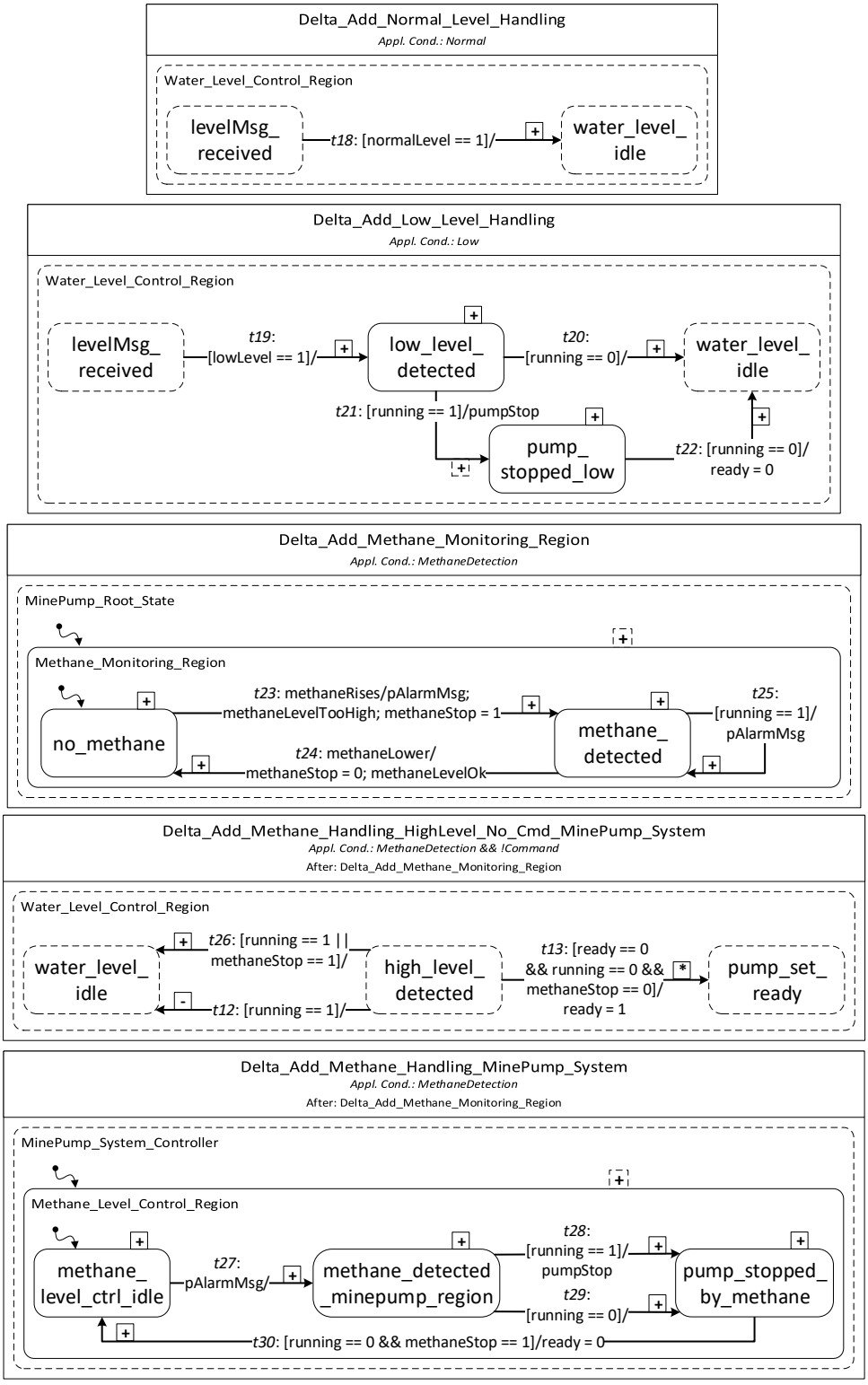

Abbildung 3.9.: Deltas der Minenpumpanlage

Die Deltas, die benötigt werden, um weitere gültige Produkte zu erstellen, sind in den Abbildungen 3.9 und 3.10 dargestellt. `Delta_Add_Normal_Level_Handling` fügt die Transition `t18` für die Behandlung einer erhaltenen Nachricht über normalen Wasserstand hinzu. `Delta_Add_Low_Level_Handling` fügt die Zustände `low_level_detected` und `pump_stopped_low` sowie die Transitionen `t19` zur Mitteilung von Niedrigwasserstand, `t20` falls die Pumpe bereits ausgeschaltet ist, `t21` zum Abschalten der Pumpe und `t22`, um bei ausgeschalteter Pumpe in den Wartezustand zurückzukehren, hinzu.

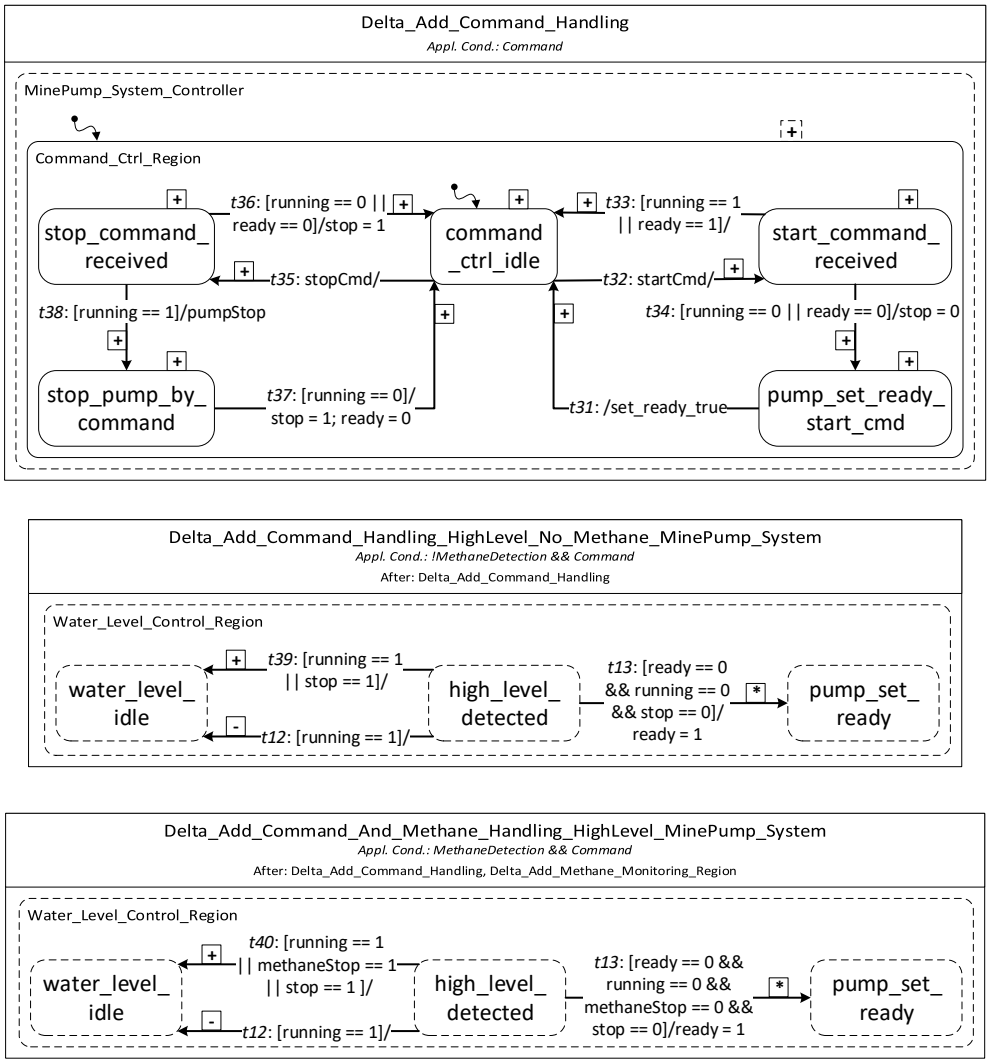

Abbildung 3.10.: Weitere Deltas der Minenpumpanlage

`Delta_Add_Methane_Monitoring_Region` fügt in `MinePump_Root_State` die `Methane_Monitoring_Region` mit den Zuständen `no_methane` und `methane_detected` hinzu, die für die Methanüberwachung zuständig sind. Ebenfalls werden die Transitionen `t23` für die Alarmnachricht bei Überschreiten des Methanschwellenwertes, `t24` für die Benachrichtigung über ein normales Methanlevel und `t25` für weitere Alarmnachrichten, falls die Pumpe trotz vorheriger Nachricht immer noch läuft, hinzugefügt.

Delta_Add_Methane_Handling_HighLevel_No_Cmd_MinePump_System ersetzt für die Methane Detection ohne Feature Command die Transition t12 durch die Transition t26, um zusätzlich auch bei einem durch Methan ausgelösten Stop ohne Aktion in den Ausgangszustand zurückzukehren. Dieses Delta kann erst nach Delta_Add_Methane_Monitoring_Region angewendet werden.

Delta_Add_Methane_Handling_MinePump_System fügt in MinePump_System_Controller die Methane_Level_Control_Region sowie deren Zustände methane_level_ctrl_idle, methane_detected_minepump_region und pump_stopped_by_methane hinzu. Darüber hinaus werden die Transitionen t27 für eine erhaltene Alarmnachricht, t28 und t29 zum Stoppen einer laufenden Pumpe beziehungsweise Überspringen einer nicht laufenden Pumpe und t30, um in den Ausgangszustand zurückzukehren, hinzugefügt. Dieses Delta kann ebenfalls erst nach Delta_Add_Methane_Monitoring_Region angewendet werden.

Delta_Add_Command_Handling wird für das Feature Command benötigt und fügt die Command_Ctrl_Region in MinePump_System_Contoller hinzu. Des Weiteren werden die Zustände command_ctrl_idle, start_command_received, pump_set_ready_start_cmd, stop_command_received und stop_pump_by_command hinzugefügt. Ebenso werden die Transitionen t31 bis t34 zum Einschalten der Überwachungsfunktion sowie t35 bis t38 zum Abschalten der Überwachungsfunktion eingefügt.

Delta_Add_Command_Handling_HighLevel_No_Methane_MinePump_System ersetzt für nicht gewähltes Feature Methane Detection und gewähltes Feature Command die Transition t12 durch t39, um zusätzlich bei ausgeschalteter Pumpe nach erkanntem Hochwasserstand in den Ausgangszustand zurückzukehren. Um dieses Delta anwenden zu können, muss vorher Delta_Add_Command_Handling angewendet worden sein.

Delta_Add_Command_And_Methane_Handling_HighLevel_MinePump_System wird bei gewählten Features Methane Detection und Command verwendet und ersetzt die Transition t12 durch t40, damit sowohl bei laufender Pumpe, ausgeschalteter oder durch Methan gestoppter Pumpe in den Ausgangszustand zurückgekehrt wird, wenn ein hoher Wasserstand erkannt wurde. Auch bei diesem Delta muss zuvor Delta_Add_Command_Handling angewendet werden.

Die Minenpumpanlage umfasst insgesamt 16 gültige Feature-Konfigurationen und folglich 16 verschiedene Produktvarianten.

3.4. Body Comfort System

Das Body Comfort System [22] beschreibt unterschiedliche Komfortfunktionen eines Fahrzeugs. Diese Fallstudie wurde ursprünglich von Müller et al. [25] entwickelt und durch Oster et al. [28] zu einer Softwareproduktlinie erweitert. Eine Übersicht über alle Features des Body Comfort Systems ist im Feature-Diagramm in Abbildung 3.11 dargestellt. Jedes Produkt der Produktlinie setzt sich mindestens aus einer Mensch-Maschine-Schnittstelle (Human Machine Interface) und einem Türsystem (Door System) zusammen. Des Weiteren kann optional ein Sicherheits-Feature (Security) dazugewählt werden.

Door System umfasst Exterior Mirror und Power Window. Beim Exterior Mirror handelt es sich durch das Mandatory-Feature Electric immer um einen elektrisch verstellbaren Außenspiegel, welcher auf Wunsch beheizbar sein kann (Heatable). Power Window wird immer mit einem Einklemmschutz (Finger Protection) geliefert und kann entweder manuell (Manual Power Window) oder automatisch (Automatic Power Window) bedienbar sein.

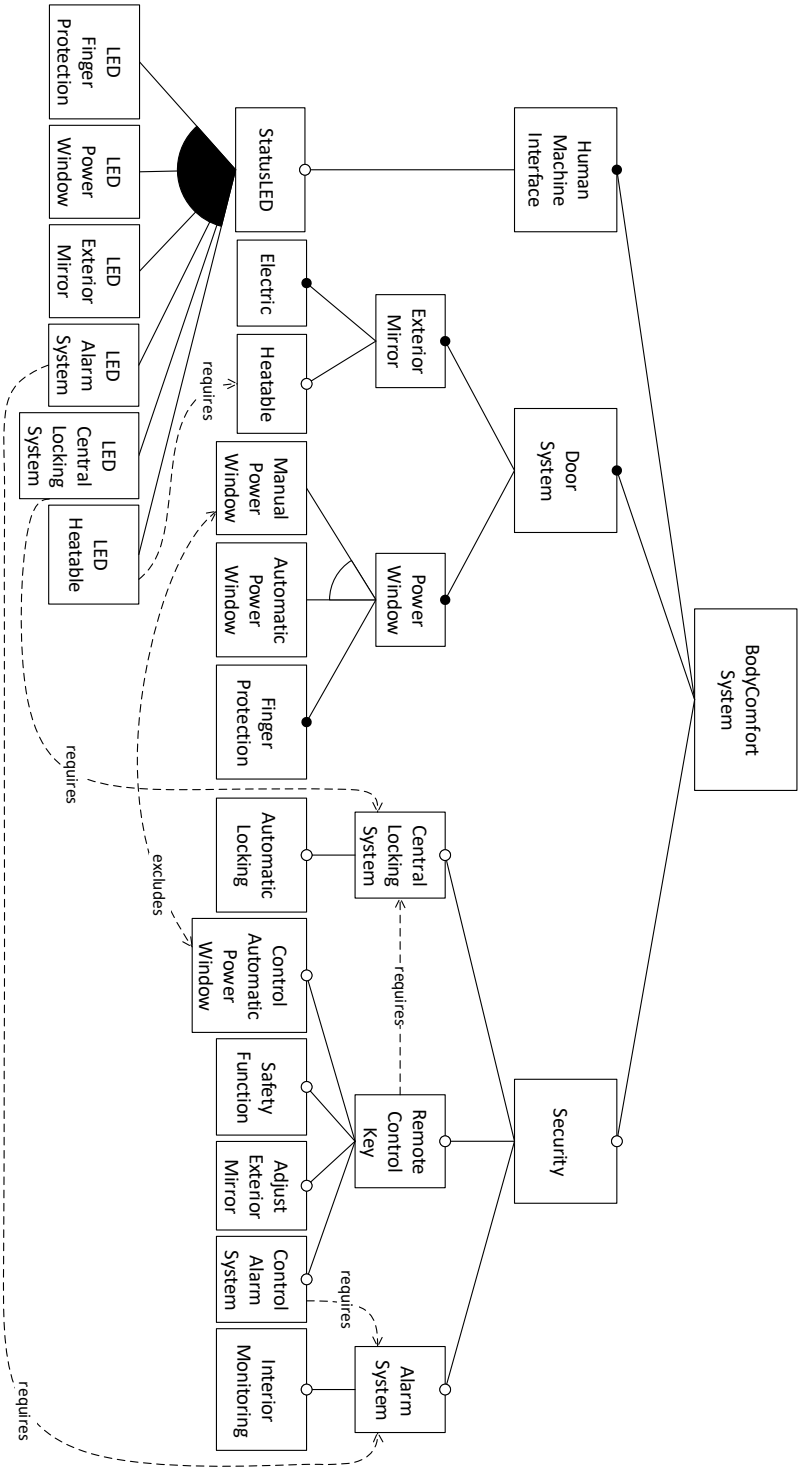

Abbildung 3.11.: Feature-Diagramm des Body Comfort Systems (nach [22])

Wurde Security selektiert, besteht die Möglichkeit, sich für eine beliebige Kombination aus den Features Central Locking System, Remote Control Key und Alarm System zu entscheiden. Ist das Central Locking System gewünscht, kann zusätzlich Automatic Locking gewählt werden. Bei Auswahl des Remote Control Key muss auf jeden Fall auch die Zentralverriegelung (Central Locking System) bestellt werden. Zusätzlich können für den Remote Control Key die Features Safety Function, Adjust Exterior Mirror, Control Automatic Power Window und Control Alarm System gewählt werden. Control Alarm System setzt voraus, dass das Feature Alarm System selektiert wird, und Control Automatic Power Window schließt ein gleichzeitiges Wählen des Manual Power Window aus. Alarm System kann ferner mit Interior Monitoring ausgestattet werden.

Sollte für das Human Machine Interface das Feature Status LED gewünscht sein, kann eine oder mehrere der LEDs LED Finger Protection, LED Power Window, LED Exterior Mirror, LED Alarm System, LED Central Locking System und LED Heatable gewählt werden. LED Alarm System benötigt die gleichzeitige Auswahl von Alarm System, LED Central Locking System ist auf das Feature Central Locking System angewiesen und für LED Heatable muss ebenfalls Heatable selektiert werden.

Verhalten

Das Body Comfort System bietet durch die beschriebenen Features verschiedene Funktionen, welche den Umgang mit einem Fahrzeug erleichtern. Durch Power Window werden die Fensterheber elektrisch gesteuert. Bei manueller Steuerung muss ein Schalter gedrückt bleiben für die Bewegung, während bei automatischer Steuerung ein einmaliges Drücken des Schalters ausreicht, um eine automatische Bewegung des Fensters auszulösen. Wird der Schalter während der Bewegung in die andere Richtung betätigt, stoppt die Bewegung. Findet während der Bewegung keine andere Aktion statt, stoppt die Scheibe automatisch, sobald sie die niedrigst- beziehungsweise höchstmögliche Position erreicht hat. Sollte die Finger Protection während der Aufwärtsbewegung, unabhängig davon, ob manuell oder automatisch gesteuert, einen Widerstand erkennen, wird die Bewegung des Fensters sofort gestoppt. LED Finger Protection leuchtet im Falle eines erkannten Objektes auf. Wurde LED Power Window gewählt, leuchtet je eine LED auf, wenn das Fenster geöffnet oder geschlossen wird, unabhängig davon, ob es sich um ein manuelles oder ein automatisches Power Window handelt. Beim Automatic Power Window leuchtet ebenfalls eine weitere LED auf, wenn das Fenster automatisch beim Abschließen hochgefahren wird.

Die Position des elektrischen Außenspiegels kann mithilfe eines Knopfes verstellt werden und bewegt sich so, je nachdem, nach links, rechts, oben oder unten. Sollte ein beheizbarer Spiegel gewählt worden sein, schaltet dieser automatisch die Spiegelheizung ein, falls die Temperatur außen zu kalt wird. Nach Ablauf einer bestimmten Zeitspanne, wird die Heizung automatisch wieder ausgeschaltet. Wurde die entsprechende LED dem Produkt hinzugefügt, zeigt diese an, wenn die Spiegelheizung eingeschaltet ist. Darüber hinaus existieren bei Feature LED Exterior Mirror LEDs, die jeweils leuchten, wenn die Spiegel links, rechts, oben oder unten die Anschlagposition erreicht haben.

Die Zentralverriegelung sorgt dafür, dass alle Türen gleichzeitig verriegelt werden, wenn mit dem Schlüssel abgeschlossen wird. Dabei kann es sich um einen mechanischen oder um einen Funkschlüssel (Remote Control Key) handeln. Ist zusätzlich auch Automatic Locking selektiert, verriegelt des Fahrzeug außerdem automatisch alle Türen, sobald es losfährt. Enthält das Produkt

die LED für die Zentralverriegelung, dann leuchtet diese, wenn das Auto verriegelt ist. Verfügt das Produkt über das Feature Automatic Power Window und sind noch Fenster geöffnet, während mit einer beliebigen Schlüsselvariante abgeschlossen wird, so werden außerdem die Fenster automatisch geschlossen.

Der Funkschlüssel setzt die Existenz der Zentralverriegelung voraus und dient dazu, das Fahrzeug auf Knopfdruck auf- oder zuzuschließen. Bei Wahl des Features Control Automatic Power Window weist der Schlüssel überdies auch Knöpfe für das automatische Öffnen und Schließen der Fenster aus der Ferne auf. Enthält der Schlüssel die Safety Function, überprüft er, ob innerhalb eines bestimmten Zeitintervalls nach Entriegeln des Fahrzeugs eine Tür geöffnet wird. Sollte dies nicht der Fall sein, verriegelt der Schlüssel das Fahrzeug automatisch wieder, um so auch die Sicherheit bei einem versehentlichen Aufschließen zu gewährleisten. Ist das Feature Control Alarm System selektiert, wird mit Verriegeln des Autos auch automatisch die Alarmanlage scharf geschaltet. Dies setzt daher ein Vorhandensein des Features Alarm System voraus.

Das Alarm System wird beim Abschließen mit dem mechanischen Schlüssel oder manuell vom Fahrer über das Human Machine Interface aktiviert. Wird der Alarm ausgelöst, dann wird ein Signal ausgegeben, welches automatisch nach einer bestimmten Zeitspanne wieder abgeschaltet wird. Der Alarm kann ebenfalls manuell über das Human Machine Interface oder bei vorhandenem Feature Control Alarm System über das Entriegeln des Fahrzeugs abgeschaltet werden. LED Alarm System lässt eine LED leuchten, wenn das Alarm System aktiviert ist. Darüber hinaus leuchtet eine LED, wenn ein Alarm ausgelöst wurde, und eine weitere als Indikation des stillen Alarms, wenn der hörbare Alarm nach Ablauf der Zeitspanne automatisch ausgeschaltet wurde. Das Alarm System kann mit Interior Monitoring erweitert werden, sodass dieses den Innenraum überwacht und einen Alarm auslöst, wenn dort eine Bewegung festgestellt wird. In diesem Fall leuchtet ebenfalls eine LED.

Das Kernmodell des Body Comfort Systems setzt sich zusammen aus Human Machine Interface (HMI), Manual Power Window (Man_PW), Finger Protection (FP) und Exterior Mirror (EM). Dargestellt ist das Kernmodell in den Abbildungen 3.12 und 3.13.

Man_PW besteht aus den Zuständen für ein geschlossenes Fenster (pw_up), ein komplett heruntergelassenes Fenster (pw_dn) und eine beliebige Position dazwischen (pw_pend). Wird der Knopf zum Öffnen des Fenster gedrückt, wenn das Fenster geschlossen (t1) oder beliebig weit geöffnet ist (t3), dann wird eine öffnende Bewegung ausgeführt. Erreicht die Scheibe die niedrigste einnehmbare Position (t5), wird trotz gedrücktem Knopf keine Bewegung mehr ausgeführt. Umgekehrt gilt, wenn der Knopf zum Schließen des Fensters betätigt wird und kein Objekt von der Finger Protection erkannt wird, dann wird, wenn das Fenster komplett (t6) oder andersweitig (t4) geöffnet ist, eine schließende Bewegung ausgeführt. Sobald das Fenster geschlossen ist (t2), bewegt es sich nicht mehr, auch wenn der Knopf zum Schließen betätigt wird. FP erkennt, wenn die Scheibe auf einen Widerstand stößt, und schaltet dann die Finger Protection ein, welche die Aufwärtsbewegung des Fensters stoppt. HMI besteht lediglich aus einem Controller.

EM setzt sich zusammen aus Zuständen, welche die Spiegelposition repräsentieren. Diese zeigen an, wenn der Spiegel sich komplett rechts, links, oben, unten oder links oben, links unten, rechts oben oder rechts unten auf Anschlag befindet. Des Weiteren repräsentiert em_pending alle Positionen dazwischen. Bewegungen, die zwischen den Extremen ausgeführt werden, werden durch Schleifen (z.B. t11 oder t30) verkörpert. Alle weiteren Transitionen bezeichnen Bewegungen, wel-

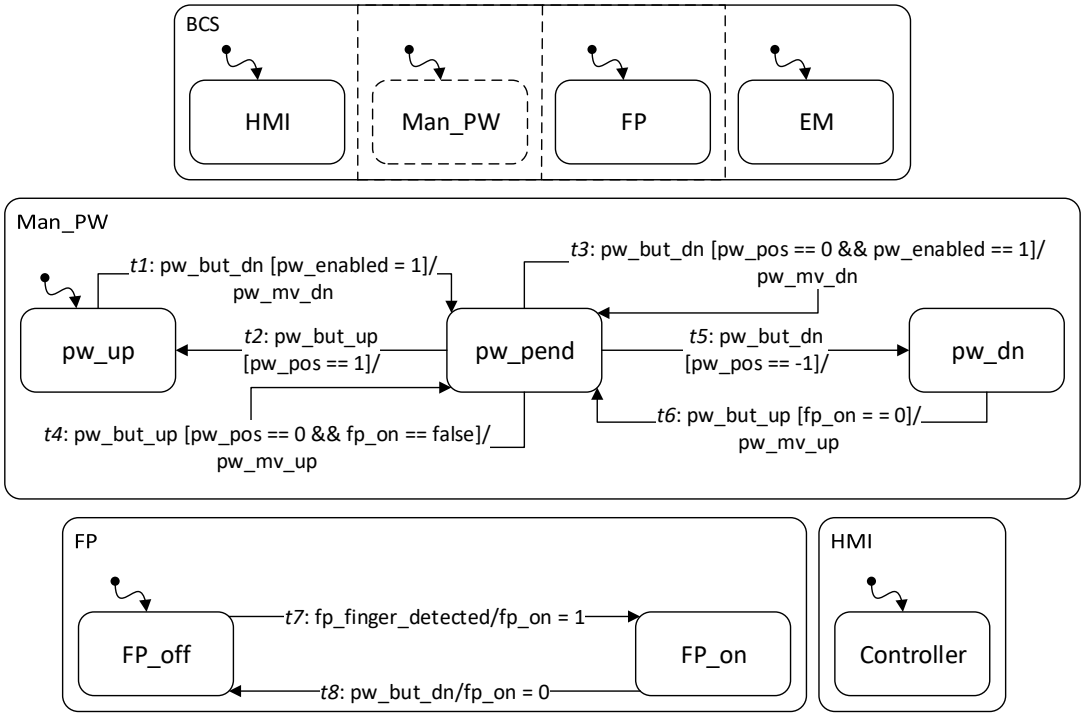

Abbildung 3.12.: Erster Teil des Kernmodells des Body Comfort Systems

che den Spiegel veranlassen, eine Extremposition zu verlassen (z.B. `t21` oder `t37`), oder welche ankündigen, dass eine Extremposition erreicht wurde (z.B. `t13` oder `t29`).

Da das Body Comfort System sehr umfangreich ist, wird in diesem Kapitel nicht die komplette Menge an Deltas aus dem Delta-Modell des BCS dargestellt, sondern lediglich einige relevante Deltas vorgestellt, welche im Verlauf von Kapitel 4.4 nützlich sein werden. Eine Übersicht über sämtliche Deltas ist bei Lity [20] zu finden. Der komplette Aufbau des Body Comfort Systems ist im 150%-Modell in Anhang A abgebildet. Das 150%-Modell, welches ursprünglich von Oster et al. [28] modelliert wurde, wurde aus Lity et al. [22] entnommen.

Vorgestellt werden hier die Deltas, welche für das Hinzufügen der LED-Statusanzeige und der Zentralverriegelung zuständig sind. Diese sind in Abbildung 3.14 dargestellt. `DAddStatusLED` fügt die leere Substate Machine `LED` hinzu. In diese werden sämtliche weitere Substate Machines eingegliedert, die für die Steuerung von LEDs zuständig sind. `DAddCLS` ergänzt die leere Substate Machine `CLS`. Durch `DAddCLSBSM` werden in `CLS` die Zustände `CLS_unlock` und `CLS_lock` sowie die Transition `t48` eingefügt. Transition `t48` entriegelt das Fahrzeug und erlaubt das Herunterfahren der Fenster, sobald eine Tür manuell durch den Schlüssel aufgeschlossen wurde. `DAddCLSBSM` darf erst nach `DAddCLS` angewendet werden.

`DAddCLSBSMManPW` wird angewendet, wenn zusätzlich das Feature `Manual Power Window` ausgewählt ist. In diesem Fall werden in `CLS` zusätzlich die Transitionen `t49` und `t50` ergänzt. Transition `t49` verriegelt das Fahrzeug, wenn mit dem Schlüssel manuell abgeschlossen wird und nicht alle Fenster geschlossen sind, und `t50` verriegelt das Fahrzeug und sperrt die Verwendung der Fensterheber, wenn mit dem Schlüssel manuell abgschlossen wird und dabei alle Fenster geschlossen sind.

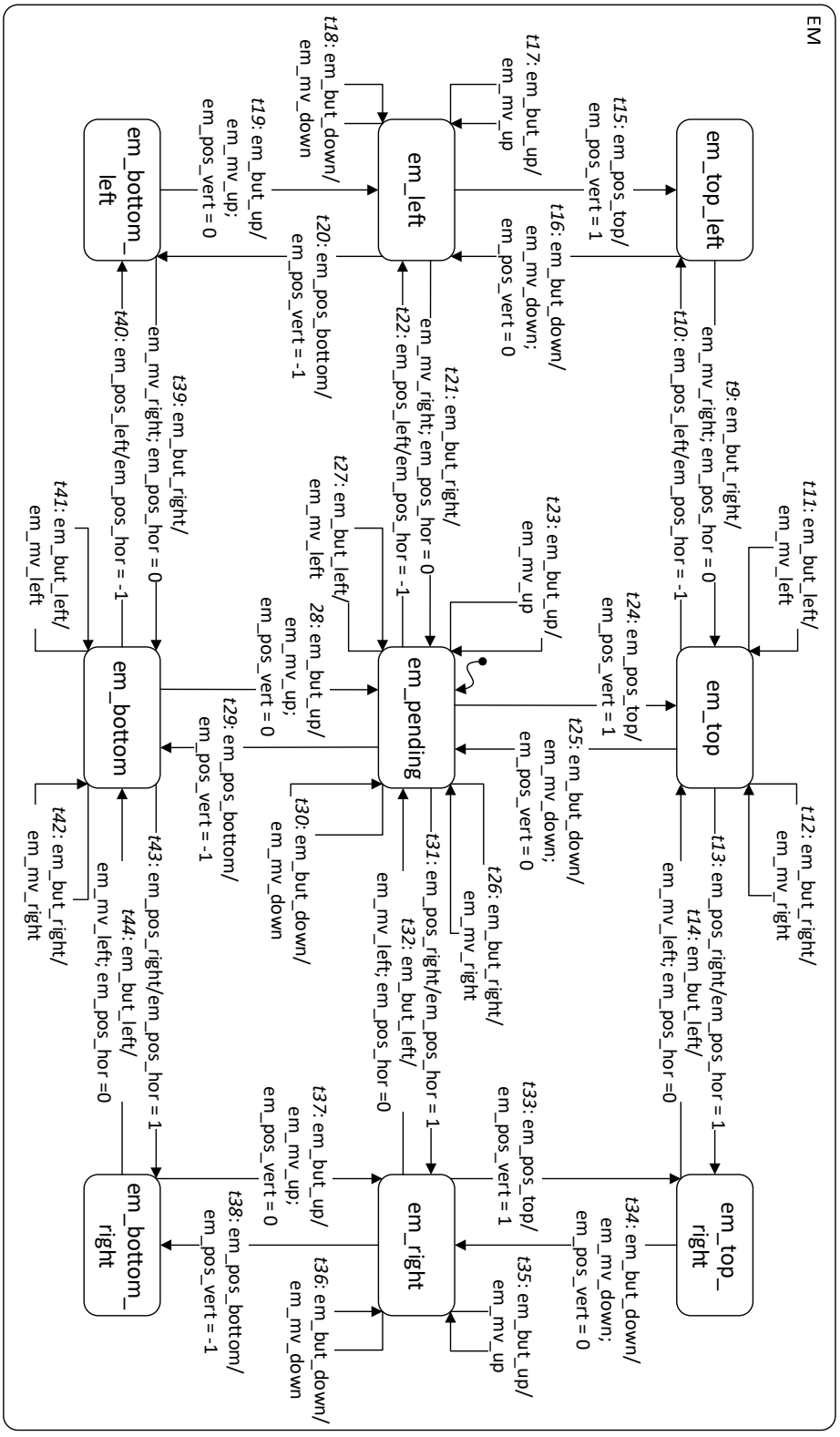

Abbildung 3.13.: Zweiter Teil des Kernmodells des Body Comfort Systems

DAddCLSBSMAutoPW ergänzt bei selektiertem Feature Automatic Power Window Transition t51, welche das Fahrzeug verriegelt und automatisch die Fenster schließt, wenn beim manuellen Abschließen mit dem Schlüssel noch Fenster geöffnet sein sollten. Gleichzeitig verhindert t51 jede weitere Nutzung der Fensterheber.

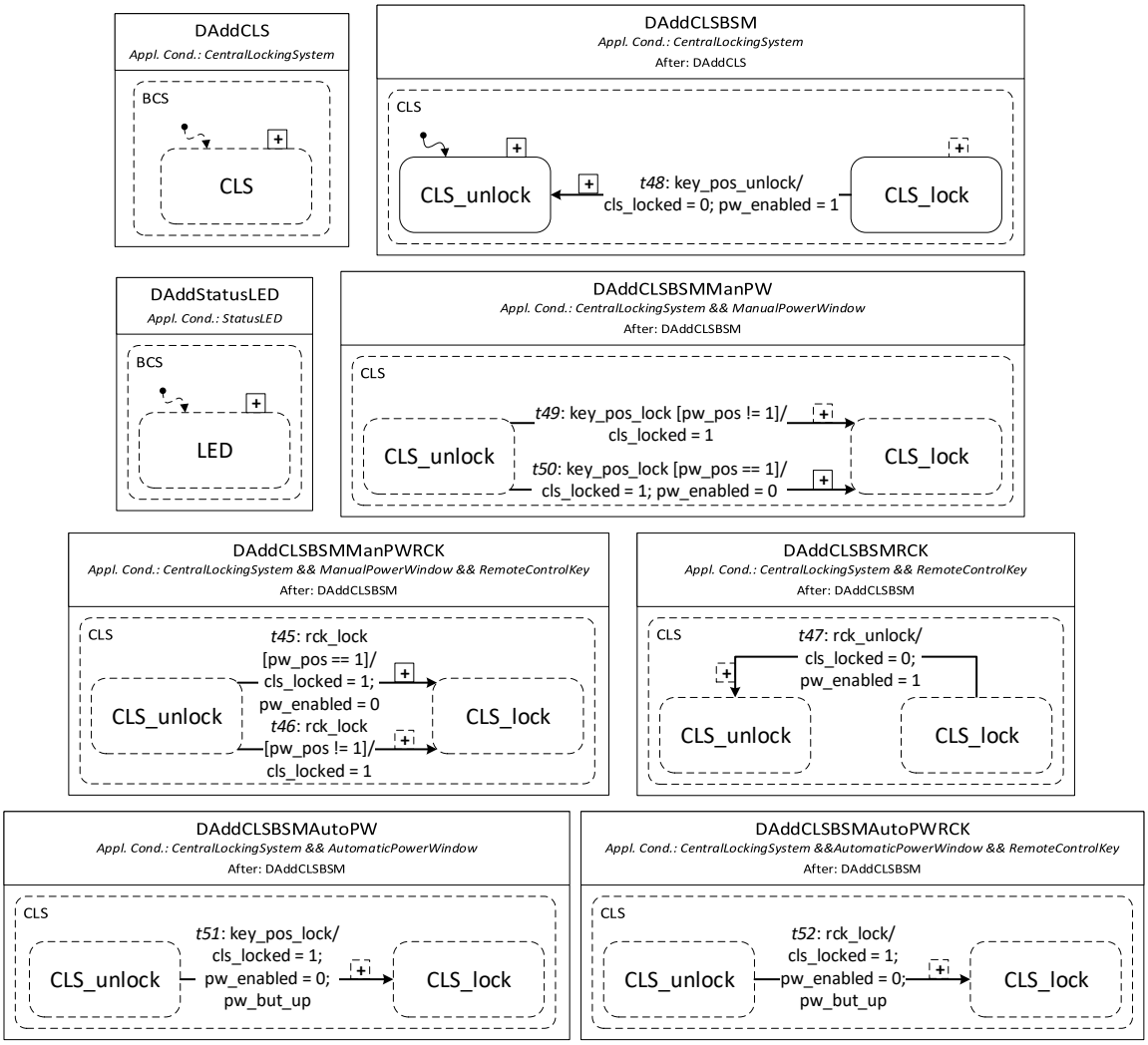

Abbildung 3.14.: Ausgewählte Deltas des Body Cofort Systems

DAddCLSBSMManPWRCK fügt in CLS die Transitionen t45 und t46 hinzu, welche für das Abschließen mit dem Funkschlüssel die gleichen Funktionen wie t49 und t50 übernehmen. Bei Verwendung von DAddCLSBSMRCK wird Transition t47 ergänzt, welche bei Abschließen durch den Funkschlüssel das gleiche Verhalten zeigt wie t48. DAddCLSBSMAutoPWRCK fügt Transition t52 ein. Diese übernimmt für das Abschließen mit Funkschlüssel die Aktionen von t51. Vor den Deltas DAddCLSBSMManPW, DAddCLSBSMAutoPW, DAddCLSBSMRCK, DAddCLSBSMManPWRCK und DAddCLSBSMAutoPWRCK muss immer zuerst DAddCLSBSM benutzt werden.

Das Body Body Comfort System umfasst mehr als 6000 Feature-Konfigurationen und Produktvarianten.

4 Evolutionsszenarien

In diesem Kapitel werden verschiedene Evolutionsszenarien zu den in Kapitel 3 beschriebenen Fallstudien entwickelt und die dadurch erfolgenden Änderungen am Verhalten dargelegt. Die Entwicklung der Szenarien erfolgt in einem zeitlichen Verlauf, gegliedert nach Fallstudien. Hierzu wird bei jedem Szenario zuerst kurz die Ausgangssituation umrissen und dann erläutert welche Änderung erfolgen soll. Die Veränderungen werden unterstützend durch höchstens ein beispielhaftes Produktmodell visualisiert, unabhängig davon, welche Anzahl an neuen Produktmodellen sich durch die Evolution ergibt. Die komplette Anzahl an neuen Varianten wird durch die Beschreibung der Änderungen im Delta-Modell abgedeckt.

4.1. Verkaufsautomat

In diesem Abschnitt werden Evolutionsszenarien für den Verkaufsautomaten beschrieben. Für den Verkaufsautomaten ergeben sich sieben Szenarien, welche die Einführung von unterschiedlichen Getränkegrößen, die variable Auswahl von Getränkegrößen, das Entfernen einer Getränkegröße, das Hinzufügen eines Milchzählers, einer Milchanzeige und eines Angebotes von Milch zu Tee und Kaffee sowie die Auswahl unterschiedlicher Milchsorten umfassen.

4.1.1. Unterschiedliche Getränkegrößen

Dieses Evolutionsszenario beschreibt das Hinzufügen einer Wahlmöglichkeit für unterschiedliche Getränkegrößen.

Beschreibung

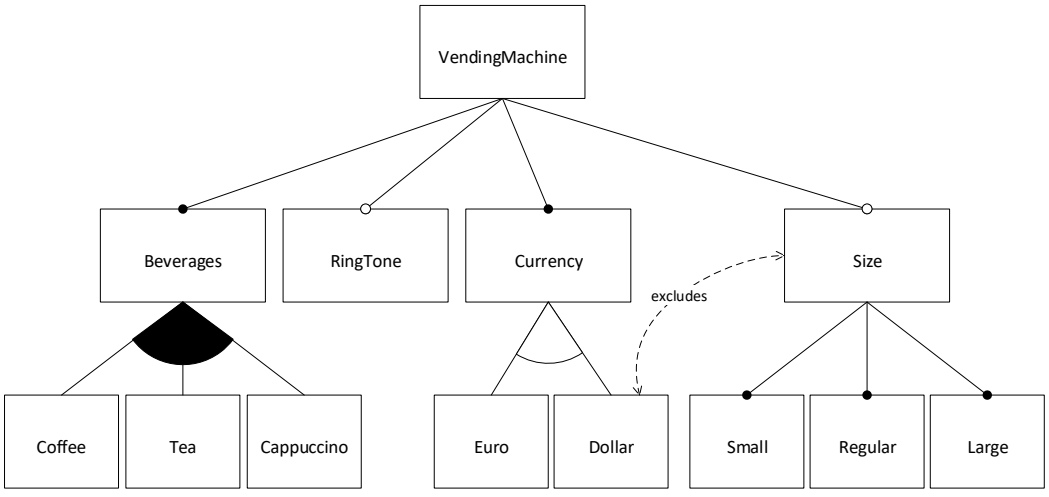

Abbildung 4.1.: Feature-Diagramm des Verkaufsautomaten mit neuem Feature `Size`

Ausgangssituation: Wie in Kapitel 3.1 beschrieben, können beim Verkaufsautomaten Kaffee, Tee und Cappuccino beliebig miteinander kombiniert werden, die Währung kann Euro oder Dollar sein und es besteht die Option einen Ton ausgeben zu lassen, wenn ein Getränk fertiggestellt ist.

Szenario: In Europa treten vermehrt Nachfragen nach unterschiedlichen Größen für die Getränke auf. Dem europäischen Automaten sollen nun also verschiedene Wahlmöglichkeiten hinzugefügt werden. Zu der normalen Standardgröße (*regular*), die bisher erhältlich war, kommen nun eine kleine (*small*) und eine große (*large*) Größe hinzu.

Umsetzung: Der Produktlinie des Verkaufsautomaten wird das Feature `Size` hinzugefügt. Da die Wahlmöglichkeit für Getränkegrößen nur bei den europäischen Automaten verfügbar sein soll, wird im Feature-Modell in Abbildung 4.1 das Optional-Feature `Size` ergänzt. Dieses steht in einer `exclude`-Beziehung mit dem Feature `Dollar`. Wenn die Wahlmöglichkeit für Getränkegrößen für ein Produkt ausgewählt wird, sollen immer alle drei Größen zur Auswahl stehen. Daher handelt es sich bei den Kind-Features `Small`, `Regular` und `Large` um Mandatory-Features.

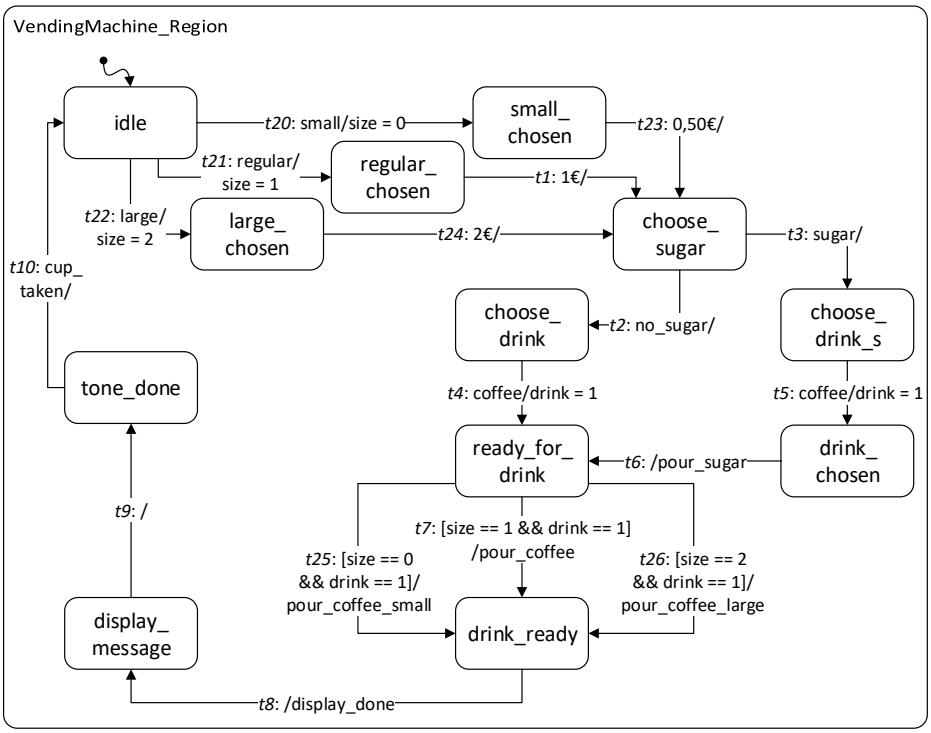

Abbildung 4.2.: Produktmodell für einen Verkaufsautomaten mit unterschiedlichen Größen

Wenn zukünftig verschiedene Größen zur Verfügung stehen sollen, müssen für diese Größen auch unterschiedliche Preise gelten. Die Standardgröße kostet weiterhin 1€, während die kleine Größe 0,50€ und die große Größe 2€ kosten werden. Es wird dabei davon ausgegangen, dass passend gezahlt werden muss.

Das entsprechende Produktmodell, das benötigt wird, um einen Automaten umzusetzen, der Kaffee in unterschiedlichen Größen anbietet, ist in Abbildung 4.2 dargestellt. Bei diesem Produktmodell wurden im Vergleich zum Kernmodell aus Abbildung 3.2 die Zustände `small_chosen`, `regular_chosen` und `large_chosen` sowie die entsprechenden Transitionen zu den Zuständen `idle` und

choose_sugar eingefügt. Darüber hinaus wurden zwischen den Zuständen ready_for_drink und drink_ready die Transitionen t25 und t26 hinzugefügt und die ursprüngliche Transition t7 wurde modifiziert, damit nun auch die Größe durch eine Bedingung beachtet wird. Bei einem Automaten, der die Getränke Tee und Cappuccino anbietet, müssten analoge Transitionen zwischen den Zuständen ready_for_drink und drink_ready eingefügt werden.

Modellierung

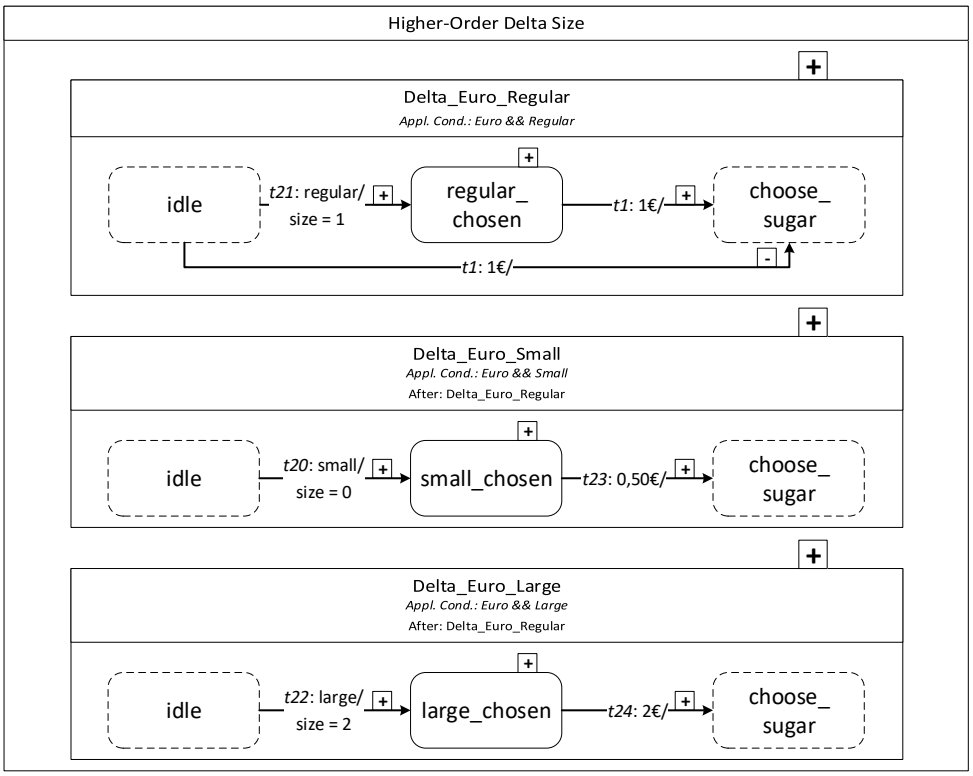

Abbildung 4.3.: Higher-Order Delta für das Hinzufügen von Deltas für unterschiedliche Getränkegrößen

Damit die in Kapitel 4.1.1 beschriebenen Automaten abgeleitet werden können, müssen sechs neue Deltas zum Delta-Modell hinzugefügt werden. Dies geschieht durch das in den Abbildungen 4.3 und 4.4 abgebildete Higher-Order Delta Size.

Um die unterschiedlichen Kosten einzuarbeiten, wird dem Delta-Modell aus Kapitel 3.1 durch *Size* ein Delta Delta_Euro_Small hinzugefügt, das für die kleine Größe den Zustand small_chosen und die Transitionen t20 und t23 ergänzt. Analog werden für die große Größe der Zustand large_chosen und die Transitionen t22 und t24 durch Delta_Euro_Large eingefügt. Für die Standardgröße werden der Zustand regular_chosen und die Transition t21 durch das neue Delta Delta_Euro_Regular hinzugefügt. Zusätzlich wird durch Delta_Euro_Regular die bereits bestehende Transition t1 zwischen den Zuständen idle und choose_sugar entfernt und zwischen dem neuen Zustand regular_chosen und choose_sugar wieder eingefügt.

Die unterschiedlichen Größen für die drei Getränke werden durch die hinzugefügten Deltas Delta_Tea_Size, Delta_Coffee_Size und Delta_Cappu_Size umgesetzt. Durch das Hinzufügen

der Transitionen `t27` und `t28` ergänzt `Delta_Tea_Size` hierzu einen kleinen und einen großen Tee. Analog werden durch `Delta_Coffee_Size` ein kleiner und ein großer Kaffee und durch `Delta_Cappu_Size` ein kleiner und ein großer Cappuccino hinzugefügt. Die bisherigen Transitionen `t15`, `t7`, `t18` und `t19` werden zusätzlich modifiziert, sodass sie Größenbedingungen berücksichtigen.

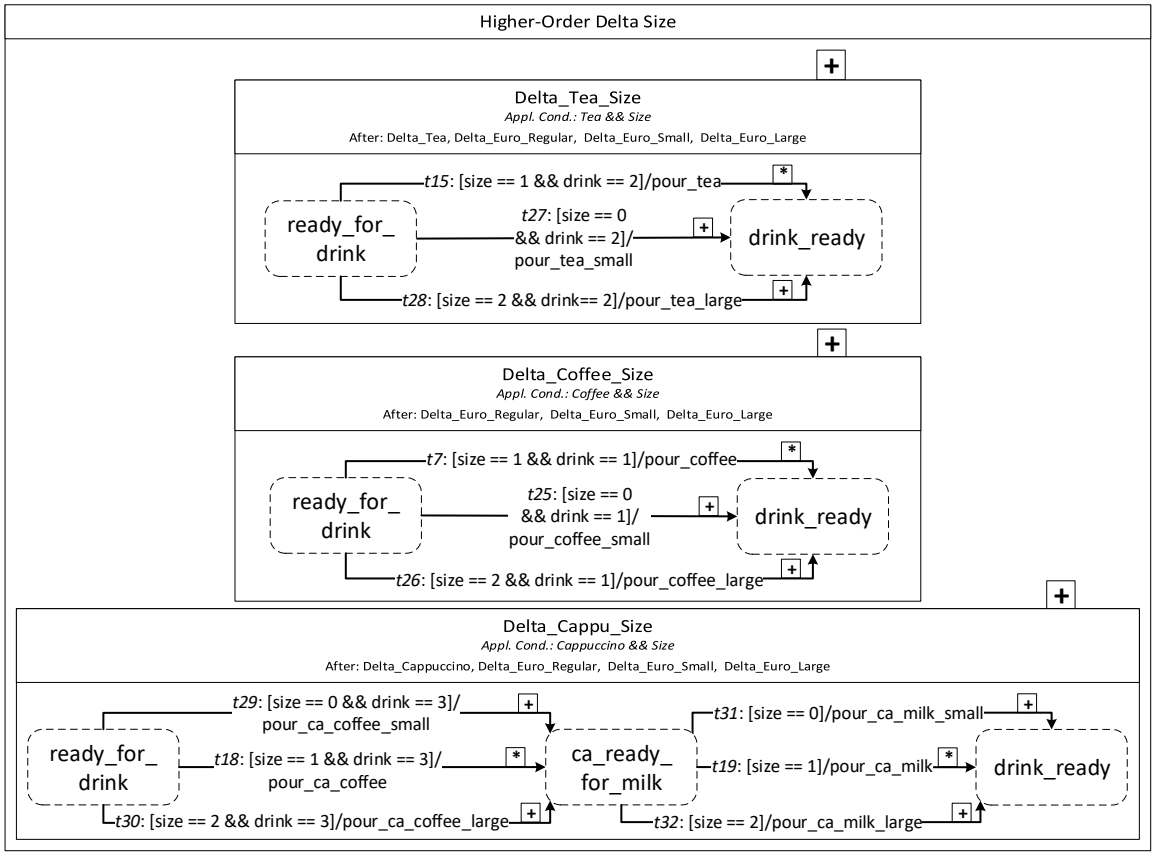

Abbildung 4.4.: Higher-Order Delta für das Hinzufügen von Deltas für unterschiedliche Getränkegrößen

Durch dieses Szenario wird der Verkaufsautomat um vierzehn neue Feature-Konfigurationen und folglich auch Produktvarianten ergänzt, sodass sich insgesamt nun je 42 Konfigurationen und Varianten ergeben.

4.1.2. Variable Getränkegrößen

In diesem Evolutionsszenario wird die Möglichkeit, die Getränkegrößen für jedes Produkt variabel zusammenzustellen, beschrieben.

Beschreibung

Ausgangssituation: Zusätzlich zu der Wahlmöglichkeit für drei verschiedene Getränke, der Alternative zwischen zwei Währungen und der optionalen Tonausgabe besteht nun die Möglichkeit unterschiedliche Getränkegrößen zu wählen. Wurde bei einem Produkt entschieden, die Wahlmöglichkeit für Getränkegrößen anzubieten, müssen zwangsläufig alle drei Größen *small*, *regular* und *large* angeboten werden.

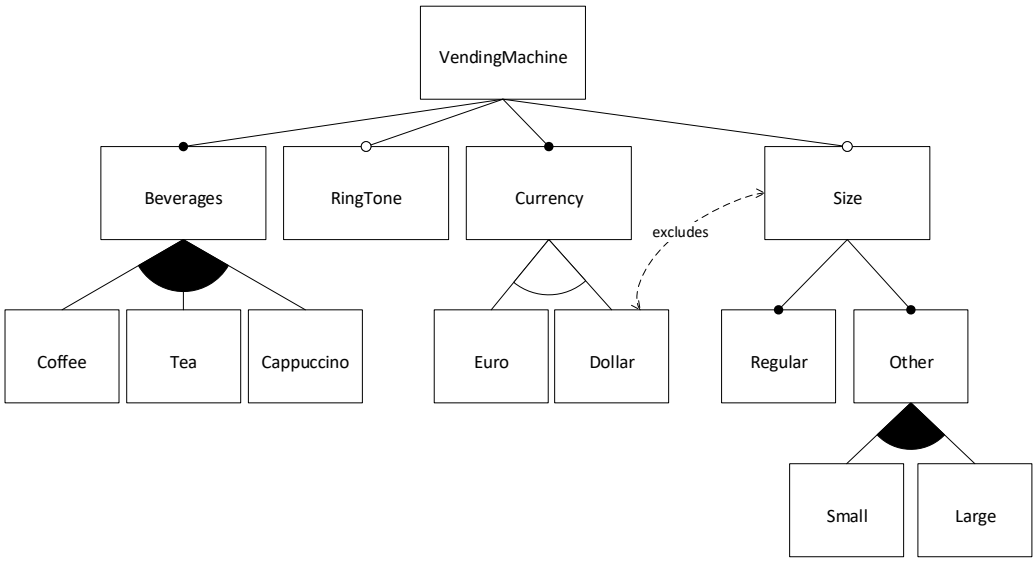

Abbildung 4.5.: Feature-Diagramm des Verkaufsautomaten mit variabler Größenauswahl

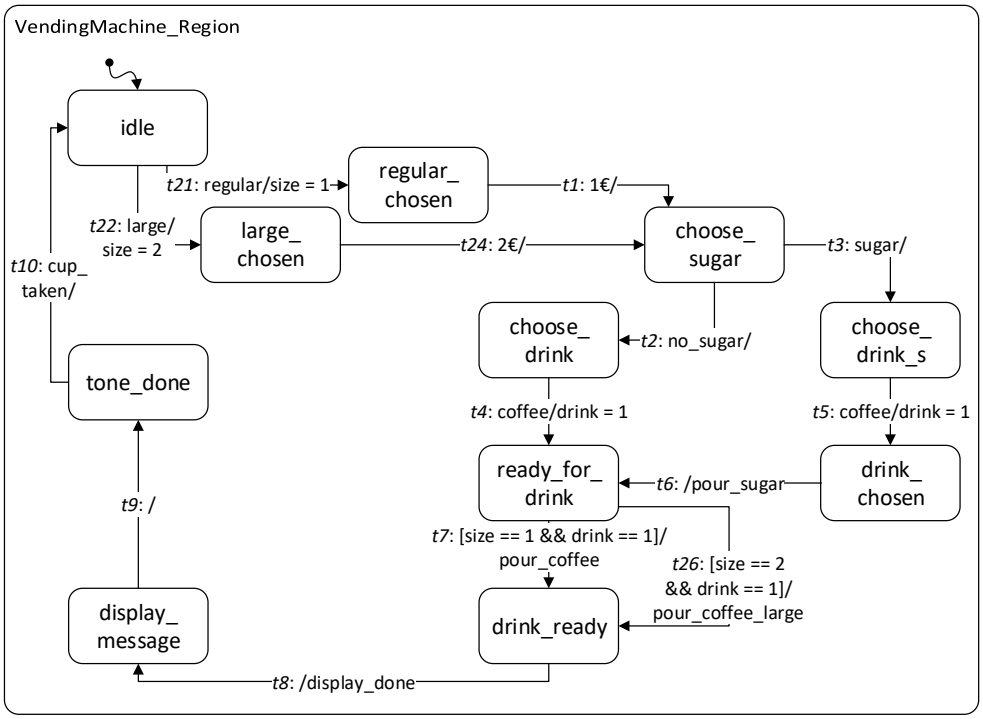

Abbildung 4.6.: Produktmodell des Verkaufsautomaten mit zwei Getränkegrößen

Szenario: Im Getränkeautomatenvertrieb kristallisiert sich heraus, dass regional unterschiedliche Vorlieben bestehen, was die Getränkegröße betrifft. So werden im Süden mehr kleine Getränke bestellt, während im Norden große Getränke favorisiert werden. Die Verkaufszahlen der Standardgröße für Getränke gleichen sich für beide Regionen ungefähr an. Die Auswahl für Getränkegrößen soll daher in Zukunft flexibler gestaltet werden. Wird die Wahlmöglichkeit für Getränkegrößen für ein Produkt ausgewählt, kann nun entschieden werden, ob die kleine Größe, die große Größe oder beide Größen angeboten werden sollen. Die Standardgröße muss allerdings immer angeboten werden.

Umsetzung: Um die Kind-Features `Small` und `Large` des Features `Size` variabel zu gestalten, werden sie von Mandatory-Features zu Or-Features umgewandelt. So ist zugleich sichergestellt, dass mindestens eines der beiden Features gewählt werden muss, wenn das Feature `Size` selektiert wird. Das Feature `Regular` bleibt unverändert und somit weiterhin ein Mandatory-Feature. So wird garantiert, dass immer mindestens zwei Getränkegrößen zur Wahl stehen. Das neue Feature-Modell ist in Abbildung 4.5 abgebildet.

Das Produktmodell in Abbildung 4.29 zeigt einen Automaten, bei dem das Feature `Size` mit den Features `Regular` und `Large` gewählt wurde. Damit ein solches Produktmodell erstellt werden kann, müssen im Vergleich zum Szenario aus Kapitel 4.1.1 die unterschiedlichen Getränkegrößen und ihre Preise getrennt voneinander eingefügt werden können.

Modellierung

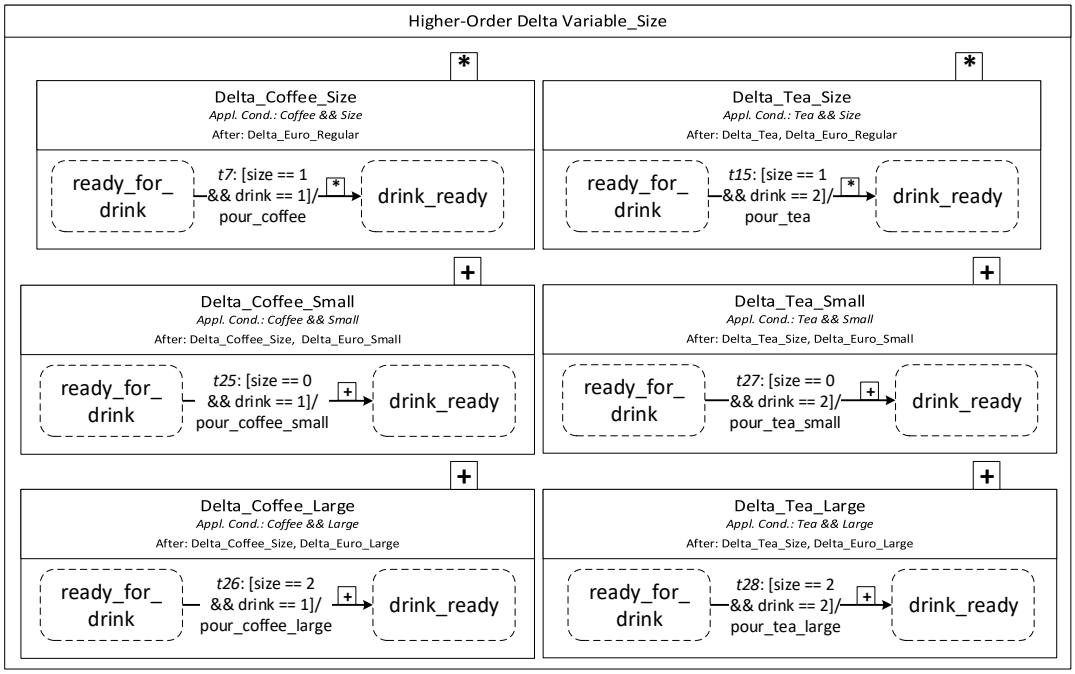

Abbildung 4.7.: Higher-Order Delta für das variable Einfügen von Getränkegrößen

In diesem Evolutionsszenario werden drei Deltas modifiziert und sechs neue Deltas zum Delta-Modell hinzugefügt. Das Higher-Order Delta `Variable_Size`, welches diese Änderungen vornimmt, ist in den Abbildungen 4.7 und 4.8 dargestellt.

Da die verschiedenen Größen zukünftig getrennt voneinander zur Verfügung stehen können, müssen die entsprechenden Deltas angepasst werden, die diese Größen einfügen. Da die Deltas `Delta_Euro_Small`, `Delta_Euro_Regular` und `Delta_Euro_Large`, welche für die unterschiedlichen Kosten zuständig sind, bereits nach Größen getrennt Änderungen vornehmen, müssen diese Deltas nicht verändert werden. Somit müssen lediglich die Deltas `Delta_Tea_Size`, `Delta_Coffee_Size` und `Delta_Cappu_Size`, welche die Getränkeausgabe in unterschiedlichen Größen einfügen, modifiziert werden. Dazu wird zuerst `Delta_Tea_Size` modifiziert, sodass dieses nur noch `t15` modifiziert. Um den Verlust der Funktion für den kleinen und großen Tee zu kompensieren, werden zwei neue Deltas `Delta_Tea_Small` und `Delta_Tea_Large` erstellt, welche die Transitionen `t27` und `t28` einfügen. Analog zu dem geschilderten Vorgehen für Tee, werden auch die Deltas `Delta_Coffee_Size` und `Delta_Cappu_Size` für Kaffee und Cappuccino modifiziert. Dementsprechend werden vier weitere neue Deltas `Delta_Coffee_Small`, `Delta_Cappu_Small`, `Delta_Coffee_Large` und `Delta_Cappu_Large` zum Delta-Modell hinzugefügt. Diese fügen `t25`, `t29` und `31`, `t26` und `30` und `t32` zum Kernmodell hinzu.

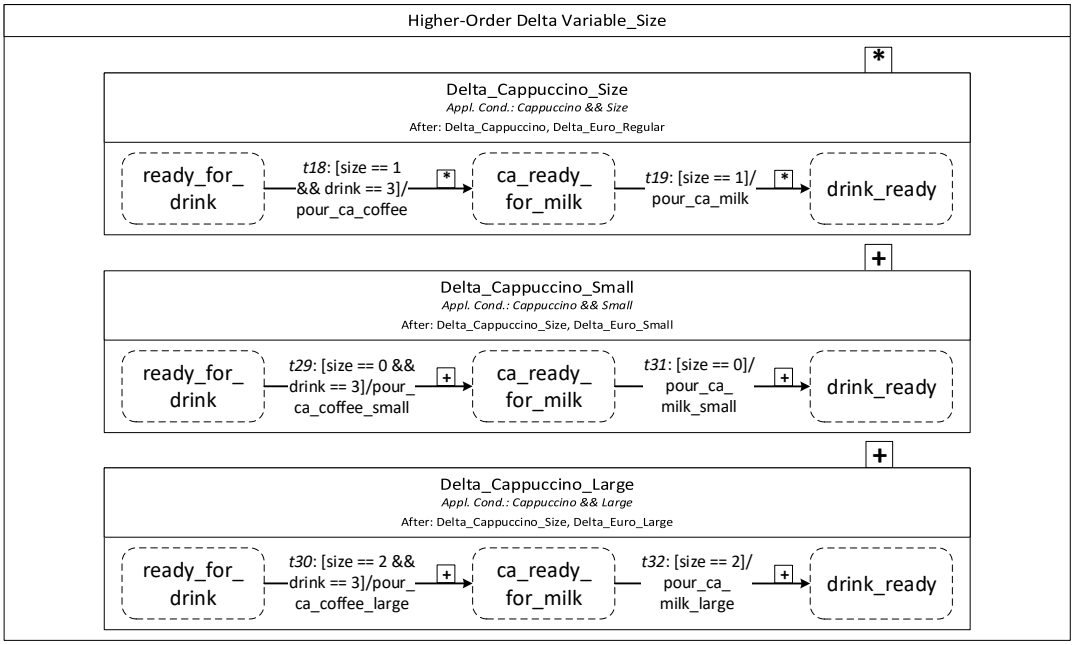

Abbildung 4.8.: Higher-Order Delta für das variable Einfügen von Getränkegrößen

Dieses Szenario fügt dem Verkaufsautomaten 28 neue Feature-Konfigurationen und Produktvarianten hinzu, wodurch sich die Gesamtanzahl beider auf 70 erhöht. Die Delta-Modifikationen haben keine Produktvarianten-Modifikationen zur Folge.

4.1.3. Entfernen einer Getränkegröße

In diesem Szenario wird das Entfernen einer Wahlmöglichkeit für Getränkegrößen beschrieben.

Beschreibung

Ausgangssituation: Es gibt weiterhin drei verschiedene Getränke zur Auswahl, es muss sich für eine von zwei Währungen entschieden werden und auf Wunsch kann ein Ton bei Getränkeausga-

be erfolgen. Darüber hinaus existiert die Möglichkeit unterschiedliche Getränkegrößen zur Wahl anzubieten. Sollten verschiedene Getränkegrößen angeboten werden, dann ist die Größe *regular* verpflichtend enthalten und es muss zusätzlich eine oder auch beide der Größen *small* und *large* gewählt werden.

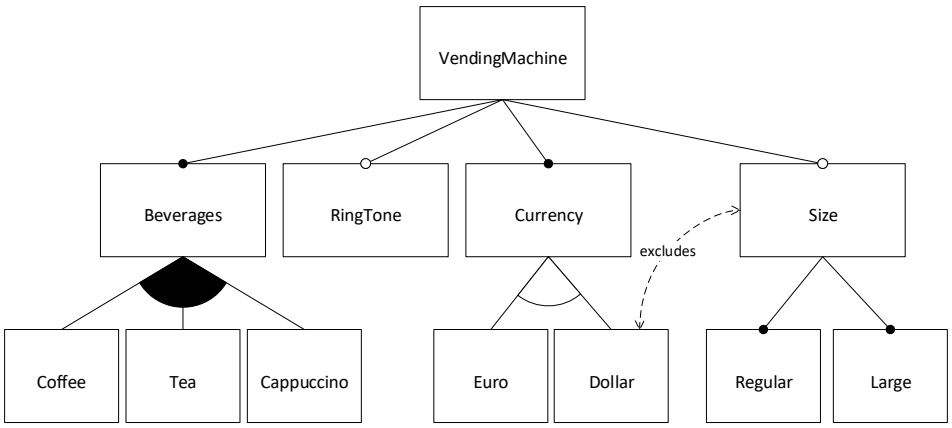

Abbildung 4.9.: Feature-Diagramm des Verkaufsautomaten ohne Getränkegröße `small`

Szenario: Verbraucherumfragen ergaben, dass nur noch verschwindend geringe Mengen an kleinen Getränken verkauft werden, sondern hauptsächlich Getränke in den Größen *regular* oder *large*. Als Folge bestellen immer weniger Getränkeautomatenbetreiber Verkaufsautomaten mit dem Angebot kleiner Getränke. Der Automatenhersteller beschließt aus diesem Grund, dass das Anbieten kleiner Getränkegrößen nicht mehr lukrativ genug ist, und nimmt diese aus der Produktlinie wieder heraus. Somit existiert zukünftig zwar weiterhin die Option, verschiedene Getränkegrößen anzubieten, jedoch stehen von nun an nur noch *regular* und *large* zur Verfügung, die dann beide verpflichtend gewählt werden müssen.

Umsetzung: Entsprechend den neuen Anforderungen an die Produktlinie wird das Feature `Small` aus dem Feature-Modell in Abbildung 4.9 entfernt. Des Weiteren ist das Feature `Large` nun ebenso wie `Regular` ein Mandatory-Feature. Auf diese Weise ist gewährleistet, dass auch eine Auswahlmöglichkeit besteht, wenn die Größenauswahl selektiert wird.

Ein Produktmodell für Verwendung der Getränkegrößen *regular* und *large* ist bereits in Kapitel 4.1.2 in Abbildung 4.29 zu sehen. Die Erstellung eines solchen Produktmodells ist also durch die bisher existierenden Deltas bereits möglich.

Modellierung

Um Produktmodelle erstellen zu können, die den neuen Anforderungen entsprechen, werden keine neuen Deltas benötigt, da bereits ausreichend Deltas zum Einfügen beziehungsweis Anpassen der Getränkegrößen *large* und *regular* vorhanden sind. Allerdings sind im Delta-Modell weiterhin Deltas vorhanden, welche die kleine Getränkegröße für die verschiedenen Getränkesorten einfügen. Diese werden fortan nicht mehr benötigt und sind daher überflüssig geworden. Aus diesem Grund werden die Deltas `Delta_Tea_Small`, `Delta_Coffee_Small`, `Delta_Cappu_Small` und `Delta_Euro_Small` durch das Higher Order Delta `Remove_Small` aus dem Delta-Modell entfernt. `Remove_Small` ist in Abbildung 4.10 dargestellt.

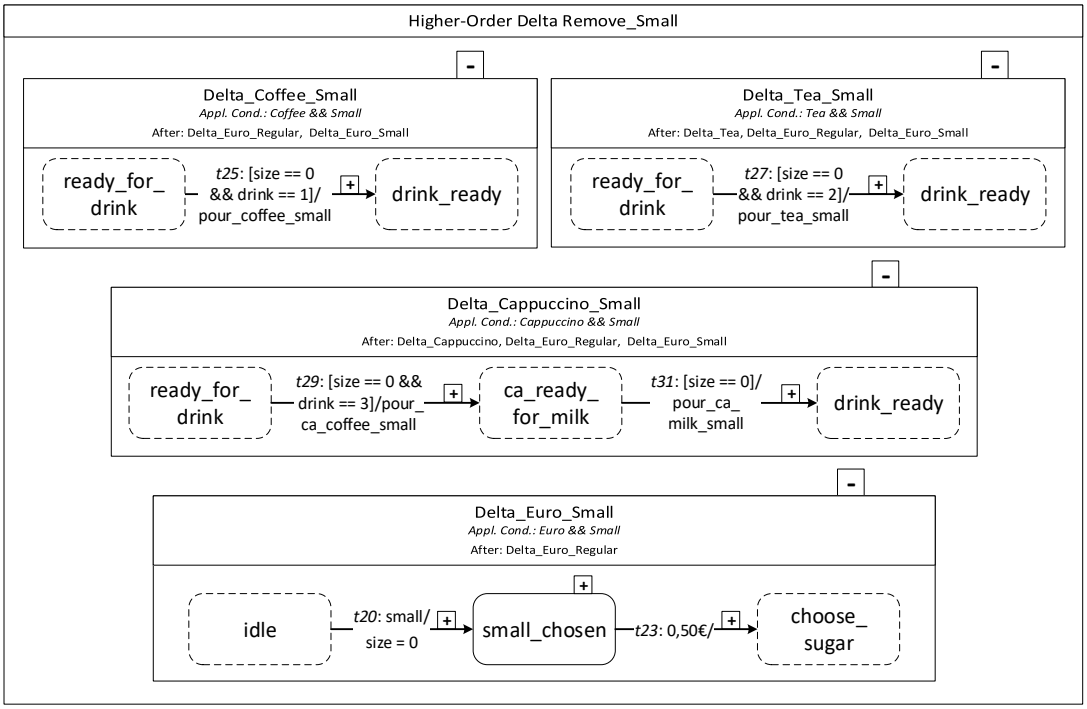

Abbildung 4.10.: Higher-Order Delta für das Entfernen der Getränkegröße *small*

Dieses Szenario verringert die Anzahl der gültigen Feature-Konfigurationen und der Produktvarianten um 28, sodass insgesamt noch je 42 Konfigurationen und Varianten übrig bleiben.

4.1.4. Milchzähler

Dieses Szenario beschreibt die Einführung eines Milchzählers, angelehnt an Kamischke et al. [16].

Beschreibung

Ausgangssituation: Für einen gültigen Automaten können eine bis drei der Getränkesorten Kaffee, Tee oder Cappuccino gewählt werden, es muss eine Währung selektiert werden und es besteht sowohl die Möglichkeit zwei Getränkegrößen anzubieten als auch einen Ton ausgeben zu lassen.

Szenario: Bei Getränkeautomaten, die Cappuccino anbieten, häufen sich Beschwerden und Reklamationen der Konsumenten über fehlerhaft ausgeschenkte Getränke. So seien häufig Cappuccinos mit zu wenig oder sogar gänzlich ohne Milch ausgegeben worden. In allen Fällen war ein leerer Milchbehälter die Ursache. Da bisher keine Möglichkeit besteht, den Füllstand eines Milchbehälters zu überprüfen, ohne umständlich den kompletten Automaten auseinanderzubauen, soll nun ein Milchzähler in jedem Automaten, der Cappuccino anbietet, eingebaut werden.

Umsetzung: Dem Feature-Modell der Produktlinie wird ein neues Feature `Milk` hinzugefügt, abgebildet in Abbildung 4.11. Da jeder Automat, der Cappuccino verkauft, ab sofort verpflichtend mit Milchzähler geliefert werden soll, besteht eine `requires`-Relation von `Cappuccino` zu `Milk`. Darüber hinaus verläuft eine `requires`-Relation von `Milk` zu `Cappuccino`. So ist sichergestellt, dass das Feature `Milk` nicht mit den Features `Coffee` oder `Tea` kombiniert werden kann, die keine Milch enthalten.

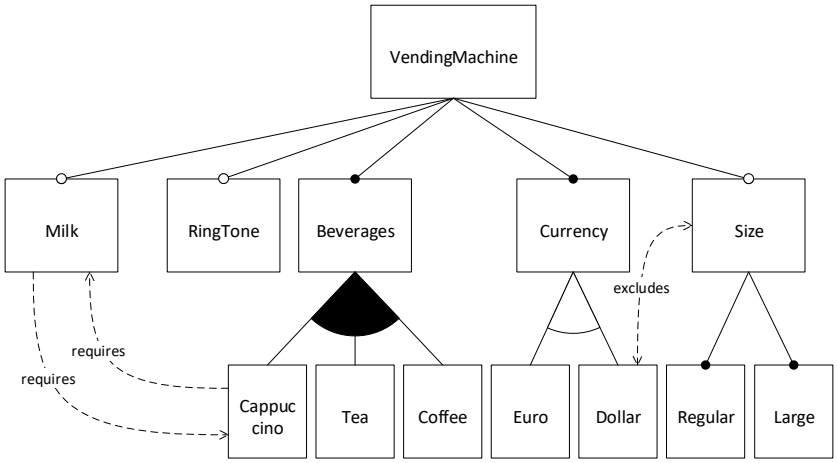

Abbildung 4.11.: Feature-Diagramm des Verkaufsautomaten mit neuem Feature `Milk`

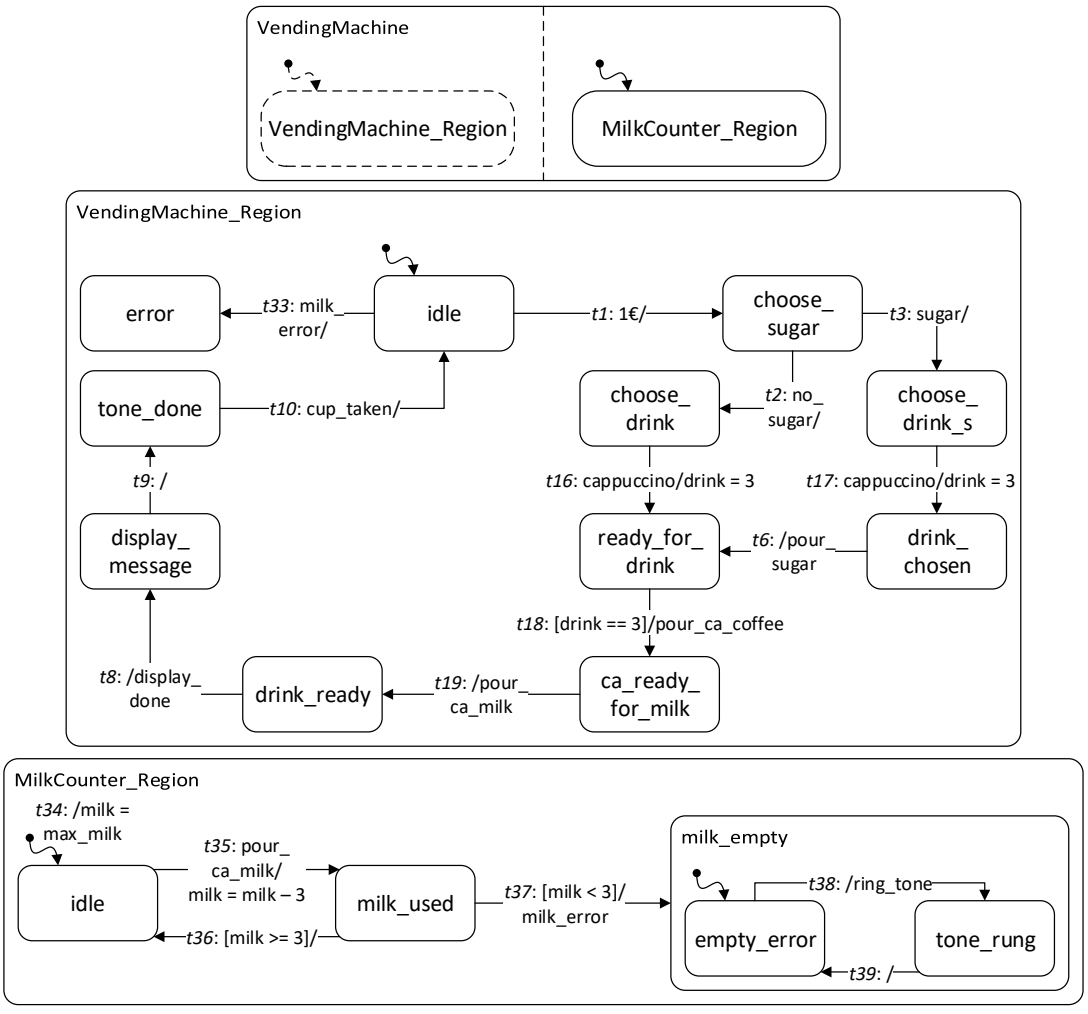

Abbildung 4.12.: Produktmodell für einen Verkaufsautomaten mit Milchzähler

In Abbildung 4.34 ist ein Produktmodell dargestellt, welches das Getränk Cappuccino in nur einer Größe samt Milchzähler anbietet. Der Milchzähler wird hierbei in einer neuen Substate Machine umgesetzt. Zu Beginn, nach Auffüllen des Automaten, wird der Milchzähler auf den maximalen Milchstand gesetzt. Jedes Mal, wenn Milch bei einem normalen Cappuccino eingegossen wird, wird der Milchzähler um die eingegossene Menge, nämlich 3 Milcheinheiten, verringert und anschließend überprüft, ob noch genug Milch für ein weiteres Getränk übrig ist. Im Falle eines Automaten, der unterschiedliche Getränkegrößen anbietet, beträgt die eingegossene Menge für einen großen Cappuccino 4 Milcheinheiten. Der Milchfüllstand wird immer mit der größten, für ein angebotenes Getränk benötigten Menge an Milcheinheiten verglichen. Sollte nicht mehr genug Milch vorhanden sein, wird ein Ton ausgegeben bis der Automat gewartet wird. Zusätzlich geht das ganze System in einen Fehlerzustand über, sodass keinerlei Getränke mehr verkauft werden können.

Modellierung

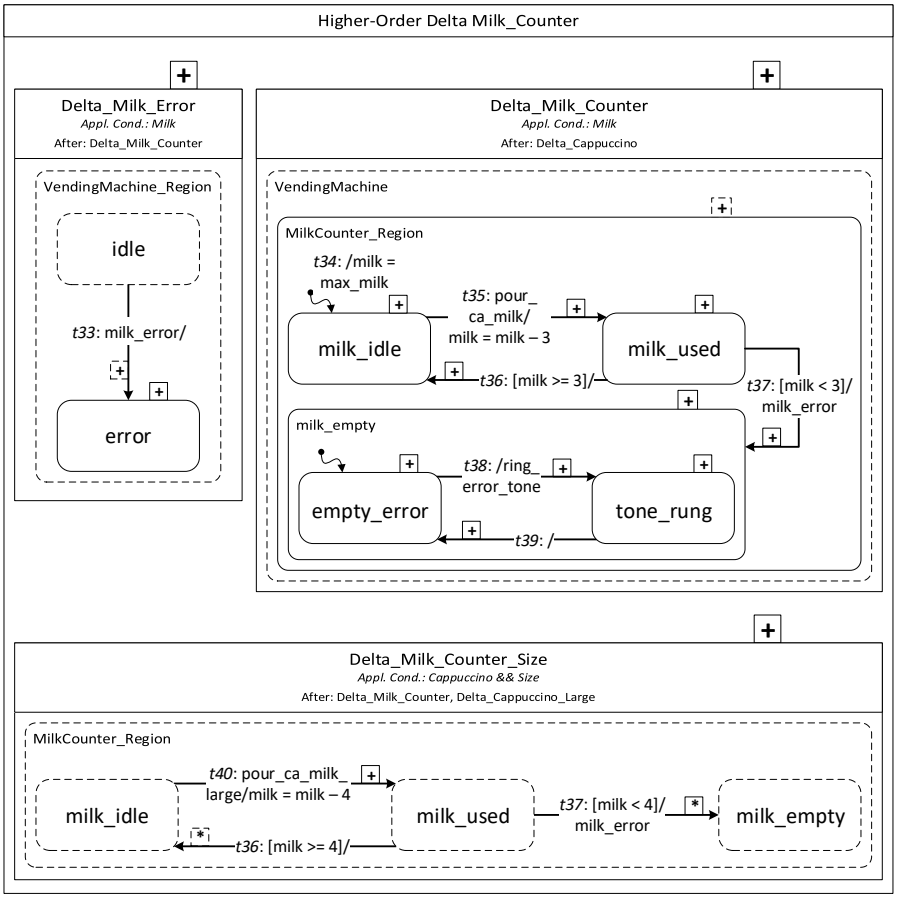

Abbildung 4.13.: Higher-Order Delta für das Hinzufügen eines Milchzählers

Die Deltas, die benötigt werden, um einen Milchzähler realisieren zu können, werden durch das Higher-Order Delta `Milk_Counter` in Abbildung 4.13 eingefügt. `Delta_Milk_Error` fügt in der `VendingMachine_Region` den Zustand `error` und die Transition `t33` hinzu. Durch `Delta_Milk_Counter` wird `VendingMachine` um die Substate Machine `MilkCounter_Region` ergänzt. Diese be-

steht aus den Zuständen `milk_idle`, `milk_used` und `milk_empty` sowie den Transitionen `t35` zum Verringern des Milchzählers, `t36` bei ausreichendem Milchstand und `t37` bei nicht ausreichendem Milchstand. Der Zustand `milk_empty` setzt sich aus den Unterzuständen `empty_error` und `tone_rung` sowie den Transitionen `t38` zur Tonausgabe und `t39` zusammen. Transition `t34` setzt beim Neustart den Milchzähler auf den Maximalwert.

`Delta_Milk_Counter_Size` findet Anwendung, wenn die Features `Milk` und `Size` in einem Produkt kombiniert werden. Dann wird in `MilkCounter_Region` die Transition `t40` zum Erhöhen des Milchzählers um 4 Milcheinheiten bei einem großen Cappuccino hinzugefügt. Zusätzlich werden die Transitionen `t36` und `t37` modifiziert, sodass der Fehlerzustand bereits ausgelöst wird, wenn nicht mehr ausreichend Milch vorhanden ist, um einen großen Cappuccino zu erstellen.

Durch das Hinzufügen eines Milchzählers zu allen Maschinen mit Cappuccino werden sämtliche Feature-Konfigurationen, die Cappuccino enthalten, ersetzt. Daher werden alle 24 Konfigurationen mit Cappuccino entfernt und 24 neue Konfigurationen ergänzt. Somit bleibt insgesamt eine Anzahl von 42 Feature-Konfigurationen. Gleiches gilt für die Produktvarianten.

4.1.5. Milchanzeige

Dieses Szenario beschreibt das Ergänzen einer Milchanzeige, angelehnt an Kamischke et al. [16].

Beschreibung

Ausgangssituation: Die Produktlinie des Verkaufsautomaten bietet die Wahl zwischen zwei verschiedenen Währungen, eine optionale Tonausgabe bei fertigem Getränk und die Möglichkeit zwei Getränkegrößen zu servieren. Es werden die Getränke Kaffee, Tee und Cappuccino offeriert und wenn Cappuccino verkauft werden soll, muss zustätzlich ein Milchzähler verwendet werden.

Szenario: Durch die Einführung des Milchzählers ergeben sich wesentliche Einbußen im Getränkegeschäft. Sobald der Milchstand zu niedrig ist, sind Automaten mit Cappuccino teils mehrere Tage außer Betrieb, bevor der Wartungsservice vorbeikommt, um den Automaten zu reinigen und wieder aufzufüllen. Aus diesem Grund soll zukünftig eine Milchanzeige am Automaten angebracht werden, die den Füllstand der Milch anzeigt, sodass der Wartungsservice rechtzeitig angefordert werden kann.

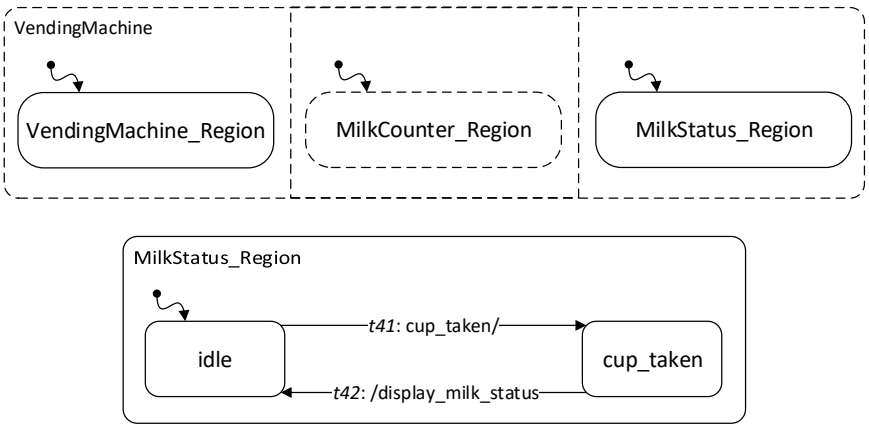

Abbildung 4.14.: Produktmodell für einen Verkaufsautomaten mit Milchanzeige

Umsetzung: Das Feature-Modell zu diesem Szenario entspricht dem Feature-Modell in Abbildung 4.11 aus Kapitel 4.1.4, da kein neues Feature für die Milchanzeige hinzugefügt wird, sondern lediglich das bestehende Feature `Milk` erweitert wird.

Zur Umsetzung der Milchanzeige müssen die Produktmodelle für Automaten mit Cappuccino um die neue Substate Machine `MilkStatus_Region` ergänzt werden. Diese ist in Abbildung 4.37 dargestellt und bewirkt jedes Mal die Anzeige des aktuellen Milchfüllstands, sobald ein Becher eines beliebigen Getränks entnommen wurde. `VendingMachine_Region` und `MilkCounter_Region` verändern sich im Vergleich zu Abbildung 4.34 nicht und werden daher in diesem Fall nicht zusätzlich dargestellt.

Modellierung

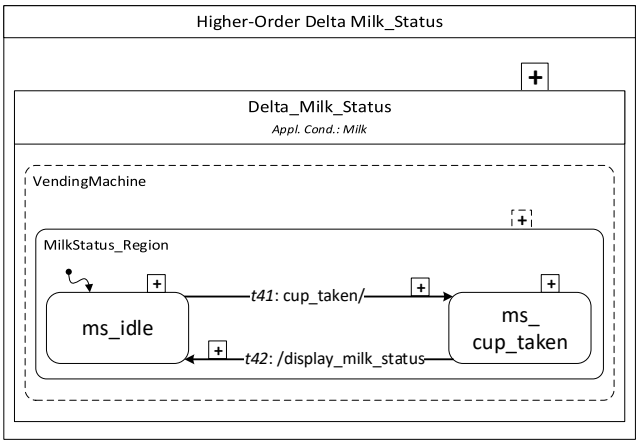

Abbildung 4.15.: Higher-Order Delta für das Hinzufügen einer Milchanzeige

Abbildung 4.15 zeigt das Higher-Order Delta `Milk_Status`, welches lediglich das Delta `Delta_Milk_Status` zum Delta-Modell hinzufügt. Dieses Delta ergänzt `VendingMachine` um die `MilkStatus_Region`, welche aus den Zuständen `ms_idle` und `ms_cup_taken` und den Transitionen `t41` zum Erkennen des entnommenen Bechers und `t42` zum Anzeigen des Milchstands besteht.

In diesem Szenario bleiben sämtliche Feature-Konfigurationen bestehen, dennoch wird die Funktion des Features `Milk` um die Milchanzeige erweitert. Dies führt dazu, dass alle 24 Produktvarianten, die Milch enthalten, modifiziert werden. Die Gesamtanzahl beträgt weiterhin 42 Feature-Konfigurationen und 42 Produktvarianten.

4.1.6. Milch obligatorisch zu Kaffee und optional zu Tee

In diesem Evolutionsszenario wird das Hinzufügen sowohl einer optionalen Wahlmöglichkeit für Milch zu Tee als auch der obligatorischen Wahlmöglichkeit für Milch zu Kaffee beschrieben.

Beschreibung

Ausgangssituation: Ein Automat benötigt eine von zwei Währungen, kann auf Wunsch einen Ton ausgeben und eine oder zwei Getränkegrößen anbieten. Er kann eine beliebige Auswahl der Getränke Kaffee, Tee und Cappuccino offerieren und wenn er Cappuccino verkauft, müssen ein Milchzähler und eine Milchanzeige benutzt werden.

Szenario: Bei Automaten, die Kaffee anbieten, tauchen vermehrt Beschwerden von Kunden auf, dass lediglich eine Auswahl für Zucker jedoch nicht für Milch zum Kaffee angeboten wird. Auch bei Automaten die sowohl Kaffee als auch Cappuccino anboten, sei Cappuccino kein hinreichender Ersatz, da die Kunden lediglich einen Schuss Milch in ihren Kaffee wollten, statt eines Kaffees der zur Hälfte aus aufgeschäumter Milch bestehe. Um diese sich häufenden Kundenwünsche zu berücksichtigen, soll zukünftig standardmäßig Milch zum Kaffee angeboten werden. Zeitgleich wurden in England und Ostfriesland Nachfragen laut, dass doch auch Tee mit Milch angeboten werden müsste. Um die regional unterschiedlichen Teevorlieben abzudecken, soll daher ab sofort optional Milch zum Tee angeboten werden. Zu Kaffee und zu Tee mit Milchfunktion müssen ebenfalls Milchzähler und Milchanzeige gestellt werden.

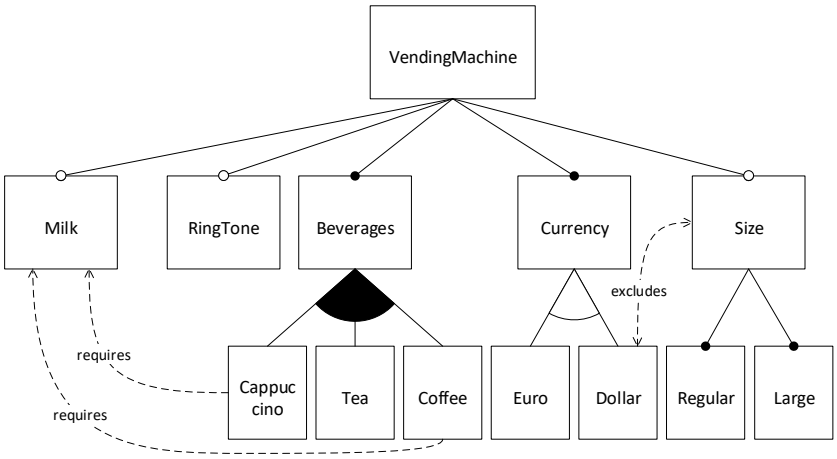

Abbildung 4.16.: Feature-Diagramm des Verkaufsautomaten mit Milch obligatorisch für Kaffee und optional für Tee

Umsetzung: Das Angebot von Milch zu Kaffee und Tee wird im Feature-Modell im bereits bestehenden Feature `Milk` umgesetzt. Wie in Abbildung 4.16 dargestellt, ergeben sich somit lediglich kleine Änderungen am Feature-Modell. Da Milch nun nicht mehr nur zu Cappuccino angeboten werden kann, entfällt die `requires`-Relation von `Milk` zu `Cappuccino`. Damit jederzeit sichergestellt ist, dass Kaffee ab sofort immer zusammen mit Milch angeboten wird, besteht nun auch vom Feature `Coffee` zum Feature `Milk` eine `requires`-Relation. Sobald Tee zusammen mit Cappuccino oder Kaffee angeboten wird, besteht automatisch die Möglichkeit zu Tee auch Milch auszuwählen.

Abbildung 4.17 zeigt ein Produktmodell, welches sowohl zu Kaffee als auch zu Tee Milch anbietet. Für beide Getränke besteht daher nach dem Eingießen des Getränks noch die Möglichkeit, zu wählen, ob Milch gewünscht ist oder nicht. Um ein solches Produktmodell zu erhalten, muss im Delta-Modell eine Maßnahme geschaffen werden, die Milchauswahl einfügen zu können. Zusätzlich wird in diesem Produktmodell der Milchzähler nach jedem Eingießen von Milch um eine Milcheinheit verringert, während aufgrund fehlenden Cappuccinos die Verringerung um drei Milcheinheiten wegfällt. Da Kaffee nicht mehr ohne Milchauswahl angeboten werden kann, können außerdem einige überflüssige Funktionen entfernt werden.

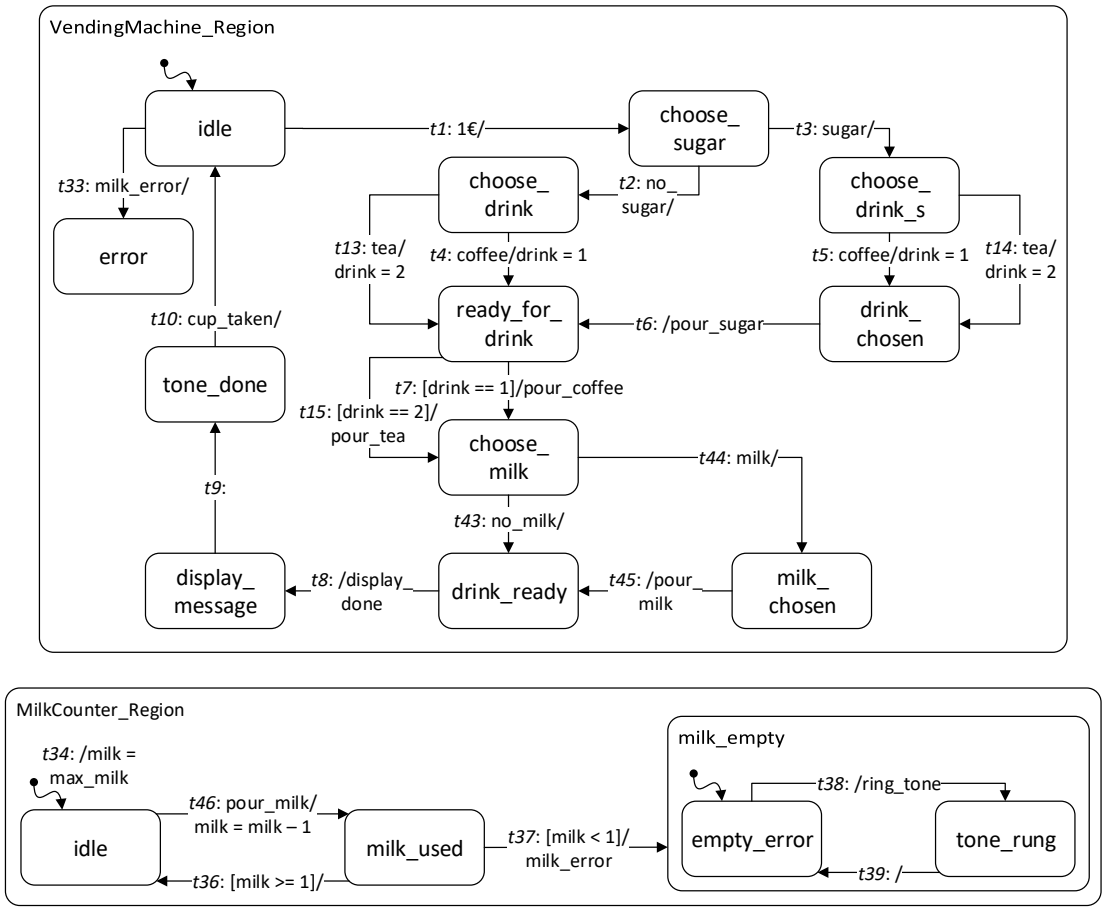

Abbildung 4.17.: Produktmodell für einen Verkaufsautomaten mit Milch zu Kaffee und Tee

Modellierung

Das Higher-Order Delta `Milk` in den Abbildungen 4.18 und 4.19 entfernt die Deltas `Delta_Coffee_Regular` und `Delta_Coffee_Large`, modifiziert die Deltas `Delta_Milk_Counter` und `Delta_Milk_Counter_Size` und fügt neun neue Deltas hinzu.

Das Einfügen einer Milchauswahlfunktion für Kaffee und Tee wird über das neue Delta `Delta_Milk` vorgenommen. `Delta_Coffee_Milk` ersetzt Transition `t7` zwischen `ready_for_drink` und `drink_ready` durch `t7` zwischen `ready_for_drink` und `choose_milk`. `Delta_Coffee_Milk` darf erst nach `Delta_Milk` angewendet werden. `Delta_Tea_Milk` übernimmt die gleiche Funktion für Transition `t15` und darf ebenfalls erst nach `Delta_Milk` verwendet werden. `Delta_Tea_Milk` findet nur dann Anwendung, wenn Feature `Size` nicht ausgewählt wird.

`Delta_Coffee_Size` und `Delta_Coffee_Large` werden aus dem Delta-Modell entfernt. Das Ersetzen und Einfügen der Transitionen `t7` und `t26` übernimmt nun `Delta_Coffee_Milk_Size`. Dieses Delta kann nur nach `Delta_Milk` und `Delta_Coffee_Milk` genutzt werden. `Delta_Tea_Milk_Size` wird angewendet, wenn das Feature `Size` selektiert ist, und verschiebt die Transitionen `t15` und `t28`. Vor diesem Delta müssen zwingend `Delta_Milk`, `Delta_Tea_Size` und `Delta_Tea_Large` verwendet werden.

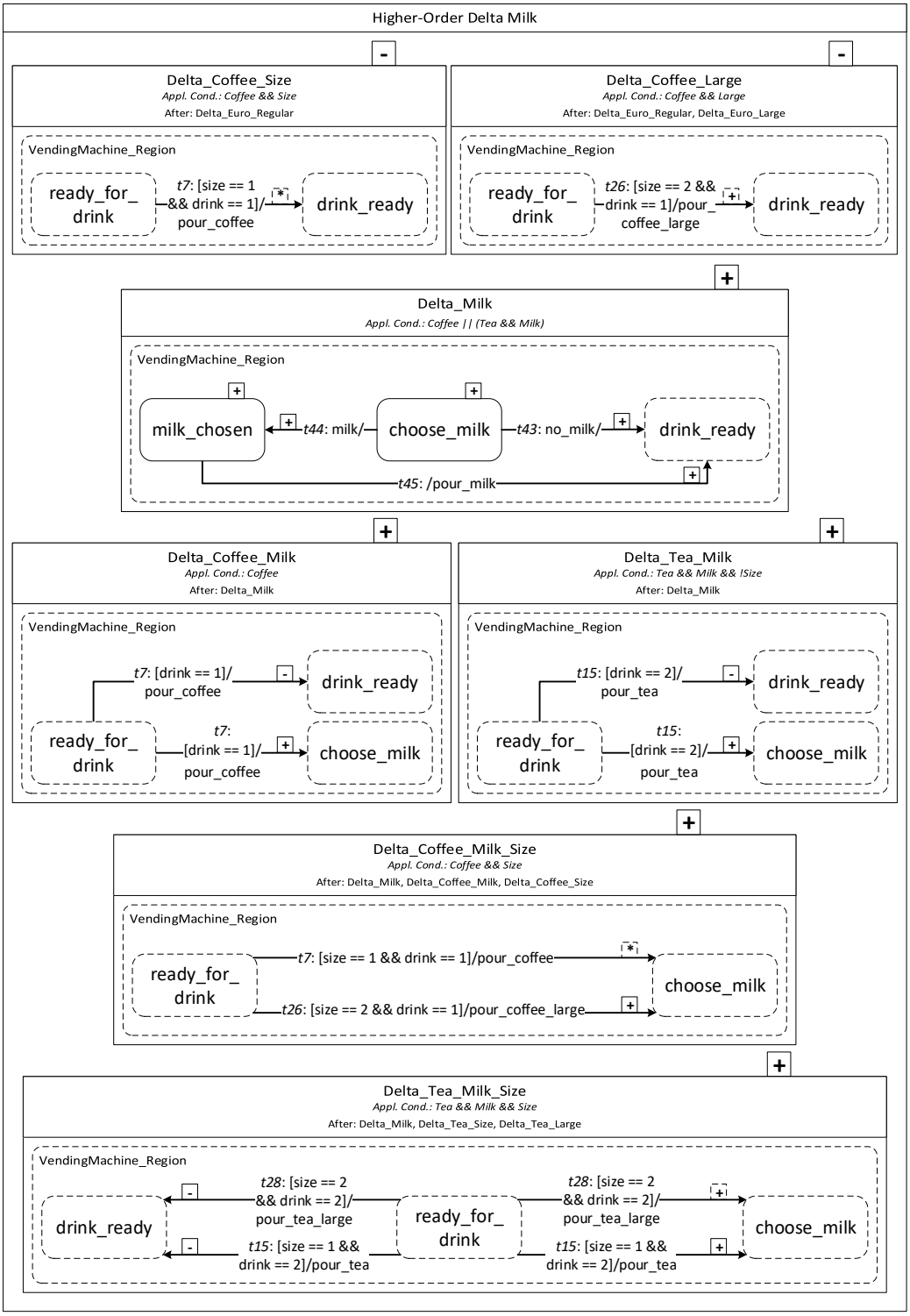

Abbildung 4.18.: Erster Teil des Higher-Order Deltas für Milch zu Kaffee und Tee

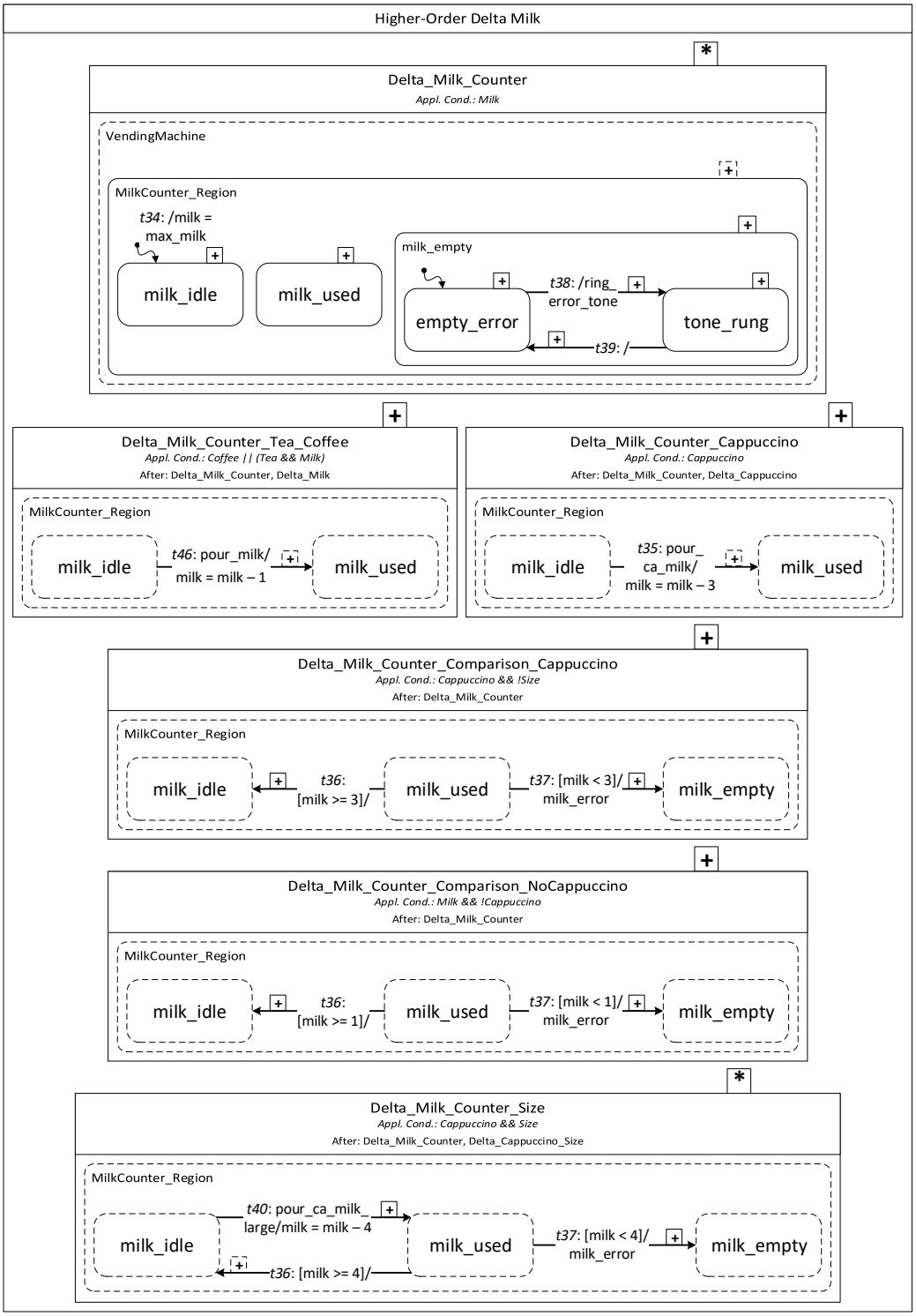

Abbildung 4.19.: Zweiter Teil des Higher-Order Deltas für Milch zu Kaffee und Tee

Um den Milchzähler für Kaffee und Tee anzupassen, wird `Delta_Milk_Counter` modifiziert. Die Transitionen `t35`, `t36` und `t37` sind auf Cappuccino spezialisiert und werden daher nicht mehr direkt durch dieses Delta eingefügt. Transition`t35` wird nun extra durch das neue Delta `Delta_Milk_Counter_Cappuccino` ergänzt, welches nur nach `Delta_Milk_Counter` eingesetzt werden darf. Die Transitionen `t36` und `t37` werden durch `Delta_Milk_Counter_Comparison_Cappuccino` hinzugefügt. Dieses Delta darf ebenfalls erst nach `Delta_Milk_Counter` benutzt werden.

Sobald Kaffee oder Tee mit Milch in einem Automaten angeboten werden sollen, wird durch `Delta_Milk_Counter_Tea_Coffee` die Transition `t46` eingefügt, die den Milchzähler nur um eine Milcheinheit verringert. Sollte es sich um einen Automaten handeln, der keinen Cappuccino anbietet, wird `Delta_Milk_Counter_Comparison_NoCappuccino` verwendet, um ebenfalls die Transitionen `t36` und `t37` hinzuzufügen. Diese Transitionen geben im Vergleich zu den Cappuccino-Transitionen `t36` und `t37` allerdings erst eine Fehlermeldung aus, wenn der verbleibende Milchfüllstand eine Milcheinheit unterschreitet statt drei.

Schließlich wird noch `Delta_Milk_Counter_Size` modifiziert, um die Anwendungsreihenfolge an die neuen und modifizierten Deltas anzupassen. Zusätzlich zu `Delta_Milk_Counter` muss nun auch `Delta_Milk_Counter_Comparison_Cappuccino` vor `Delta_Milk_Counter_Size` angewendet werden.

Durch dieses Szenario werden 18 neue Feature-Konfigurationen ergänzt, während 12 Konfigurationen entfernt werden. Analog werden dadurch 18 Produktvarianten hinzugefügt und 12 Varianten entfernt. Darüber hinaus werden 18 bestehende Produktvarianten modifiziert. Dies ist in erster Linie auf neu hinzugefügte Deltas zurückzuführen. Insgesamt ergeben sich somit je 48 gültige Feature-Konfigurationen und Produktvarianten für den Verkaufsautomaten.

4.1.7. Unterschiedliche Milchsorten

Dieses Evolutionsszenario beschreibt das Erweitern der Milchfunktion um eine Auswahl an verschiedenen Milchsorten.

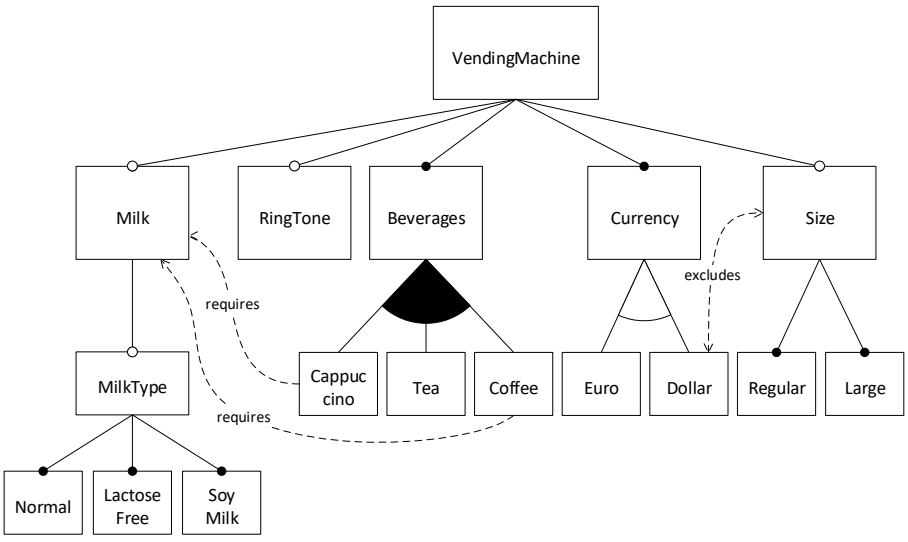

Abbildung 4.20.: Feature-Diagramm des Verkaufsautomaten mit unterschiedlichen Milchsorten

Beschreibung

Ausgangssituation: Die Produktlinie des Verkaufsautomaten bietet zwei verschiedene Währungen, die Möglichkeit bei fertigem Getränk einen Ton auszugeben, zwei unterschiedliche Getränkegrößen und die Getränke Kaffee, Tee und Cappuccino. Cappuccino enthält standardmäßig Milch, zu Kaffee muss immer Milch angeboten werden und zu Tee kann Milch angeboten werden. Sobald Milch zum Getränk gehört, sind ein Milchzähler und eine Milchanzeige inbegriffen.

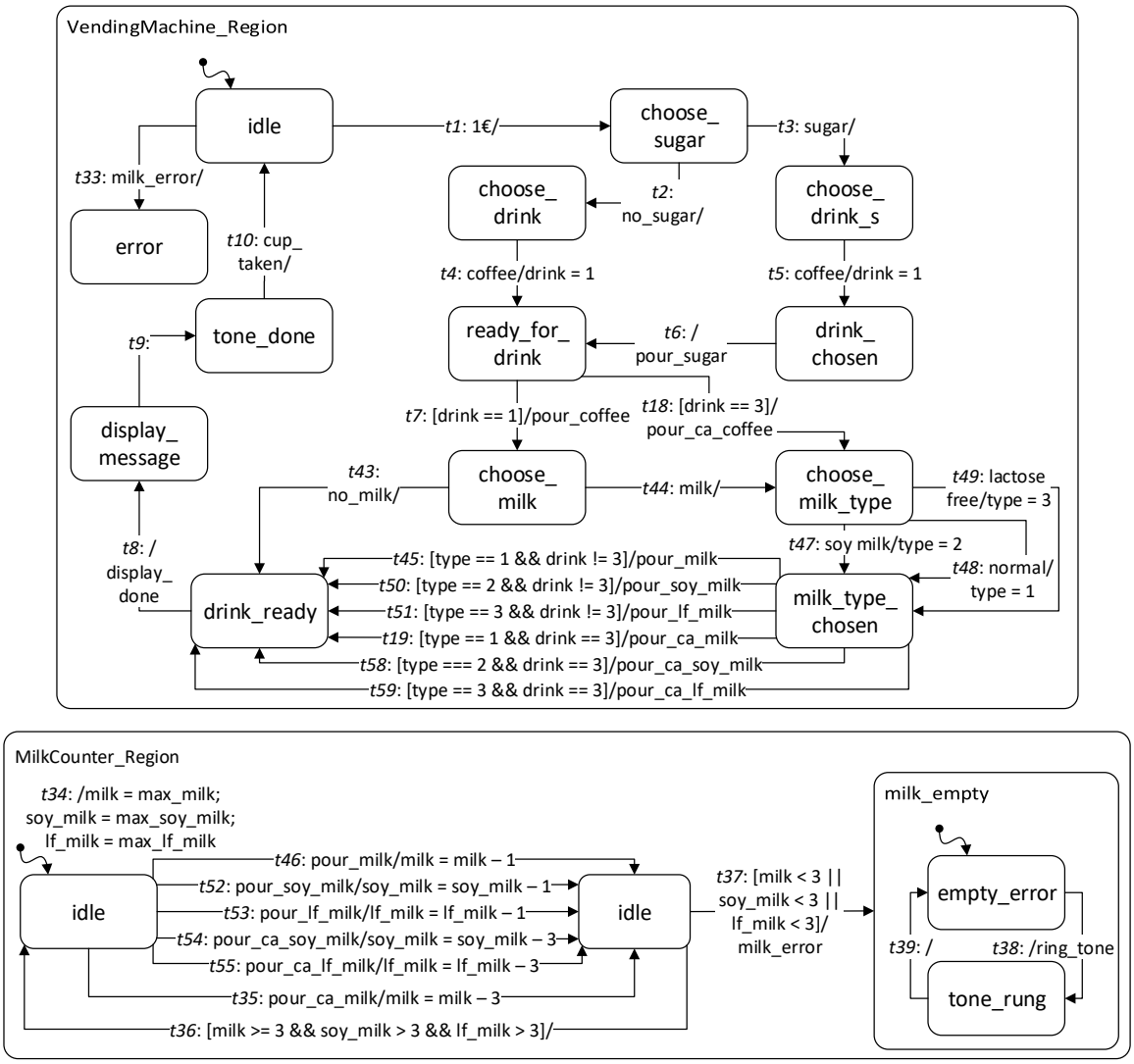

Abbildung 4.21.: Produktmodell für einen Verkaufsautomaten mit unterschiedlichen Milchsorten

Szenario: Die Einführung eines Milchangebots bei Kaffee und Tee hat sich als durchschlagender Erfolg herausgestellt. Aufgrund von sich immer weiter verbreitenden Zivilisationskrankheiten wie Laktoseintoleranz und Veganismus entschließen sich die Automatenhersteller daher, das Milchangebot für die Automaten zu erweitern. Für laktoseintolerante Personen soll ab sofort laktosefreie Milch und für Veganer Sojamilch angeboten werden können. Normale Vollmilch bleibt weiterhin im Angebot erhalten.

Umsetzung: Zur Umsetzung der Auswahl von unterschiedlichen Milchsorten wird das neue Optional-Feature `Milk Type` in das Feature-Diagramm in Abbildung 4.20 eingefügt. `Milk Type` besitzt die drei Mandatory-Features `Normal`, `Lactose Free` und `Soy Milk` als Kind-Features, somit müssen alle Milchsorten angeboten werden, sobald das Feature `Milk Type` selektiert wird. Das Feature wirkt sich automatisch auf alle Getränkesorten aus, die für den Automaten gewählt wurden.

Abbildung 4.21 zeigt ein Produktmodell mit Kaffee und Cappuccino und den drei Milchsorten zur Auswahl. Nachdem sich der Kunde bei Kaffee für Milch entschieden hat, muss er nun auch wählen, welche Milchsorte er möchte. Bei Cappuccino wird nach dem Eingießen des Kaffees automatisch abgefragt, welche Milch gewünscht ist. Im Milchzähler werden die einzelnen Füllstände der unterschiedlichen Sorten um eine beziehungsweise drei Milcheinheiten verringert, je nachdem ob Kaffee oder Cappuccino serviert wurde. Sobald die höchste Menge, die für ein Getränk benötigt wird (in diesem Fall drei Milcheinheiten) für eine der Milchsorten unterschritten wird, wechselt der Automat in den Fehlerzustand. Um diesen Ablauf umsetzen zu können, müssen der bisherige Milchauswahlprozess sowie der Milchzähler angepasst und erweitert werden.

Modellierung

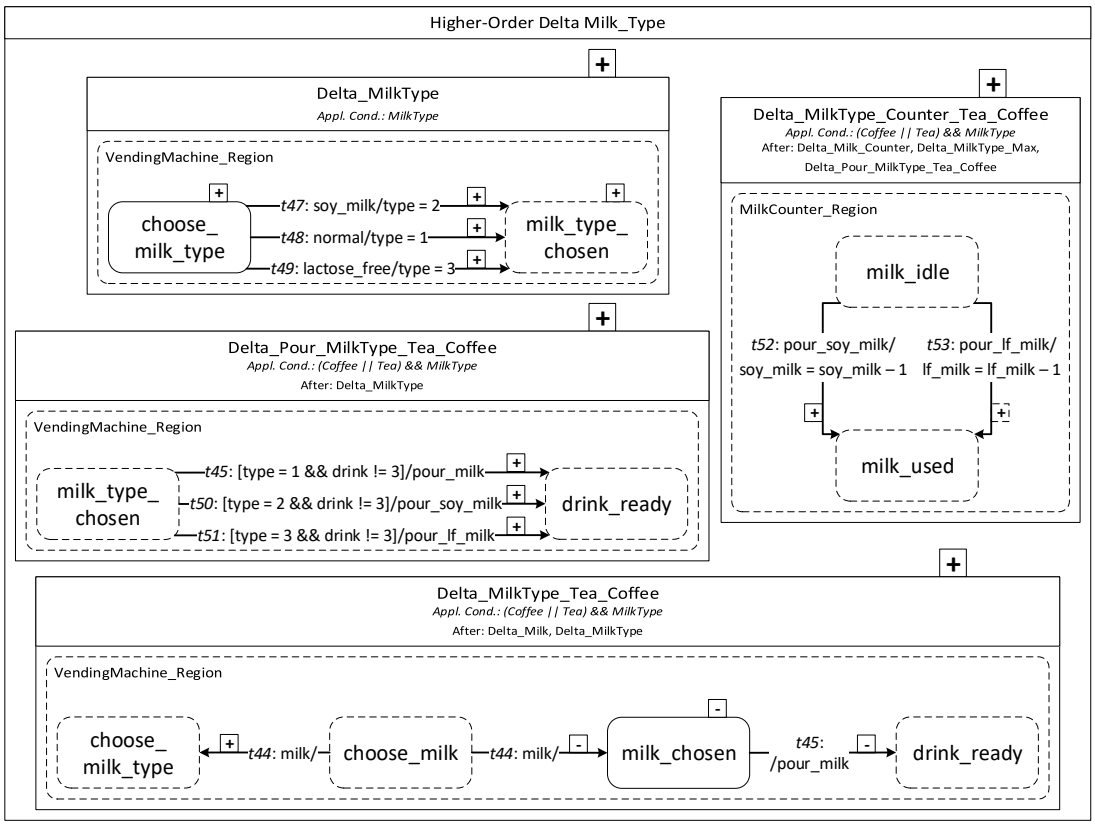

Abbildung 4.22.: Erster Teil des Higher-Order Deltas für unterschiedliche Milchsorten

Das Higher-Order Delta `Milk_Type` fügt insgesamt sechzehn neue Deltas zum Delta-Modell hinzu und modifiziert ein bereits vorhandenes Delta, um die Milchauswahl an die neuen Milchsorten anzupassen. Es ist in den Abbildungen 4.22, 4.23, 4.24 und 4.25 abgebildet.

Delta_MilkType fügt die Zustände choose_milk_type und milk_type_chosen sowie die Transitionen t47, t48 und 49 zum Wählen des Milchtyps zum Modell hinzu. Delta_Pour_MilkType_Tea_Coffee ergänzt die Transitionen t45, t50 und t51, um die gewählte Milchsorte für Tee oder Kaffee einzufüllen und kann erst nach Delta_MilkType angewendet werden. Delta_MilkType_Tea_Coffee fügt t44 zwischen choose_milk_type und choose_milk hinzu und entfernt den Zustand milk_chosen und die bereits anderweitig eingefügten Transitionen t44 und t45. Hierfür müssen zuvor Delta_Milk und Delta_MilkType angewandt worden sein. Delta_MilkType_Counter_Tea_Coffee ergänzt die MilkCounter_Region um die Transitionen t52 und t53 zum Hinabsetzen des Zählers für laktosefreie und Sojamilch bei Kaffee oder Tee. Dieses Delta darf erst nach den Deltas Delta_Milk_Counter, Delta_MilkType_Max und Delta_Pour_MilkType_Tea_Coffee verwendet werden.

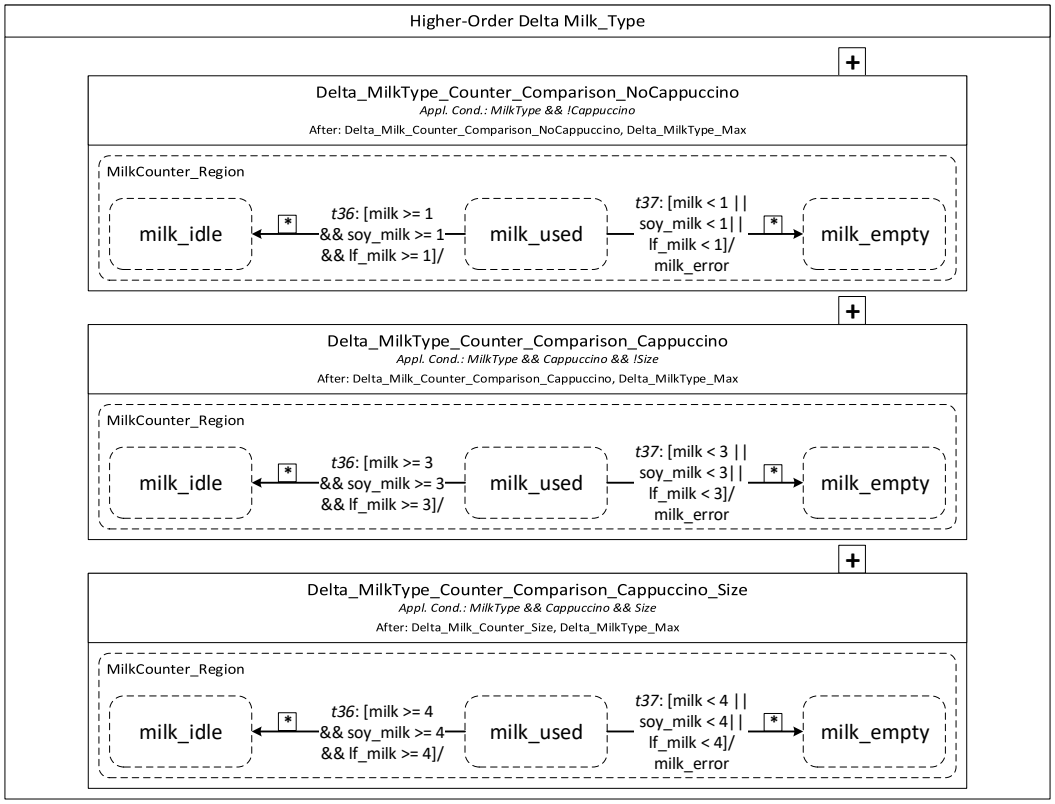

Abbildung 4.23.: Zweiter Teil des Higher-Order Deltas für unterschiedliche Milchsorten

Delta_MilkType_Counter_Comparison_NoCappuccino, Delta_MilkType_Counter_Comparison_Cappuccino und Delta_MilkType_Counter_Comparison_Cappuccino_Size modifizieren je die Transitionen t36 und t37 damit auch die Füllstände für laktosefreie und Sojamilch mit dem jeweiligen, zulässigen Niedrigstand verglichen werden. Delta_MilkType_Counter_Comparison_NoCappuccino darf daher erst nach Delta_Milk_Counter_Comparison_NoCappuccino, Delta_MilkType_Counter_Comparison_Cappuccino nach Delta_Milk_Counter_Comparison_Cappuccino und Delta_MilkType_Counter_Comparison_Cappuccino_Size nach Delta_Milk_Counter_Size angewendet werden. Zusätzlich muss vor allen drei Deltas Delta_MilkType_Max verwendet werden.

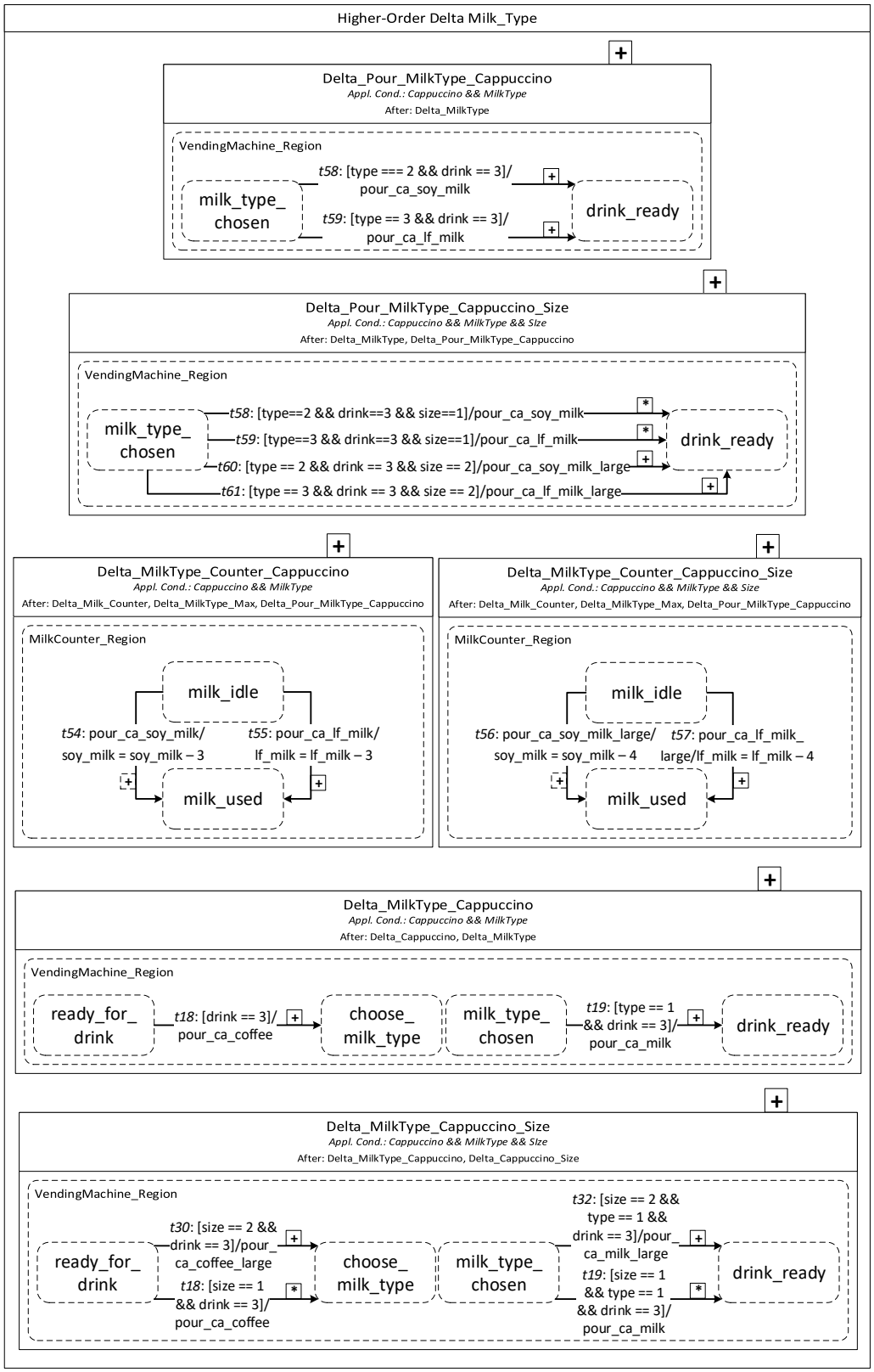

Abbildung 4.24.: Dritter Teil des Higher-Order Deltas für unterschiedliche Milchsorten

Delta_Pour_MilkType_Cappuccino fügt die Transitionen t58 und t59 zum Einfüllen von laktosefreier und Sojamlich bei Cappuccino ein und kann nur nach Delta_MilkType angewendet werden. Delta_Pour_MilkType_Cappuccino_Size ergänzt die Transitionen t60 und t61 und modifiziert die Transitionen t58 und t59 und darf folglich erst nach Delta_MilkType und Delta_Pour_MilkType_Cappuccino verwendet werden.

Delta_MilkType_Counter_Cappuccino fügt die Transitionen t54 und t55 für die Milchzählerverringerung von laktosefreier und Sojamilch bei normalem Cappuccino zum Modell hinzu. Delta_MilkType_Counter_Cappuccino_Size übernimmt diese Aufgabe für den großen Cappuccino durch Ergänzen von t56 und t57. Bei beiden Deltas müssen zuvor die Deltas Delta_Milk_Counter, Delta_MilkType_Max und Delta_Pour_MilkType_Cappuccino verwendet werden.

Delta_MilkType_Cappuccino fügt die Transitionen t18 zwischen ready_for_drink und t19 zwischen milk_type_chosen und drink_ready ein. Dieses Delta darf erst nach den Deltas Delta_Cappuccino und Delta_MilkType angewendet werden. Delta_MilkType_Cappuccino_Large ergänzt die Transitionen t30 und t32 und modifiziert t18 und t19, um deren Übergangsbedingungen an die Getränkegröße anzupassen. Folglich müssen vor Delta_MilkType_Cappuccino_Large zuerst Delta_MilkType_Cappuccino und Delta_Cappuccino_Size angewendet werden. Damit t30 und t32 nicht zweimal an unterschiedlichen Stellen eingefügt werden, wird Delta_Cappuccino_Large in seiner Anwendungsbedingung so modifiziert, dass es nicht benutzt werden kann, wenn das Feature Milk Type selektiert ist.

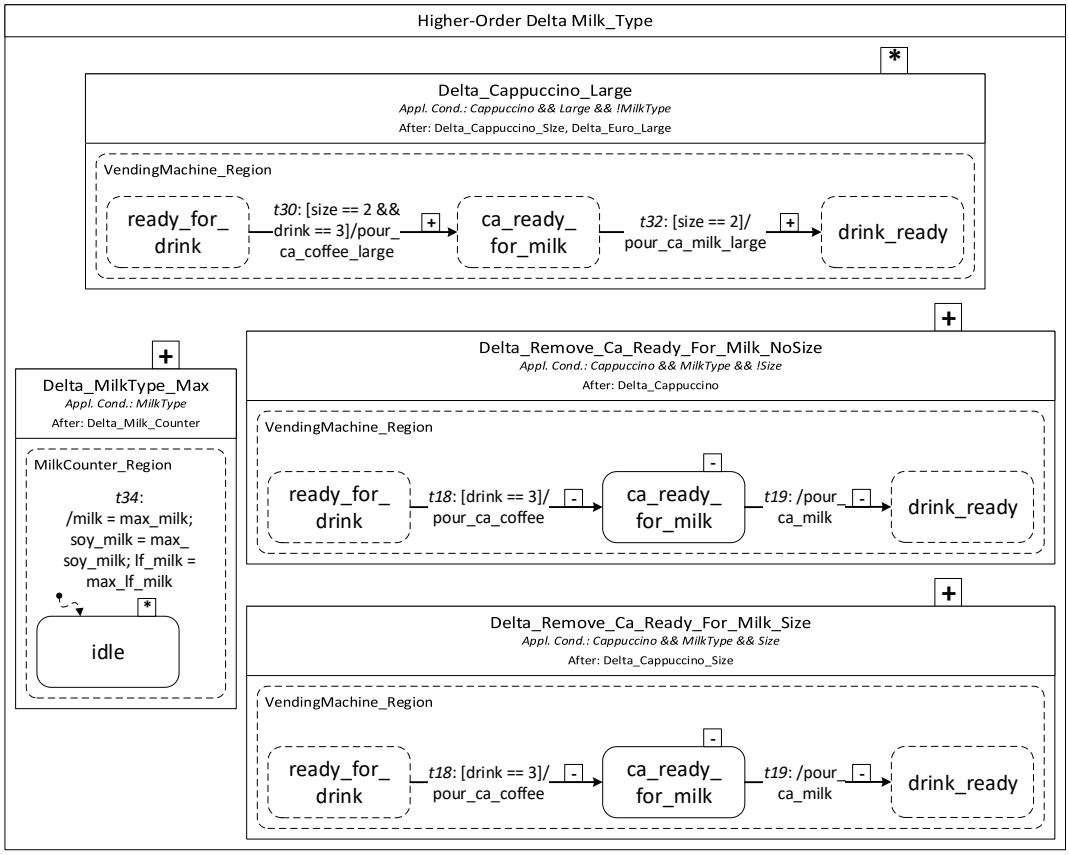

Abbildung 4.25.: Vierter Teil des Higher-Order Deltas für unterschiedliche Milchsorten

Delta_MilkType_Max modifiziert t34, sodass bei einem Neustart des Automaten zusätzlich auch Sojamilch und laktosefreie Milch auf den Maximalfüllstand gesetzt werden. Delta_MilkType_Max kann erst nach Delta_Milk_Counter angewendet werden. Die Deltas Delta_Remove_Ca_Ready_For_Milk_NoSize und Delta_Remove_Ca_Ready_For_Milk_Size entfernen beide den Zustand ca_ready_for_milk und die Transitionen t18 und t19. Die beiden Deltas unterscheiden sich lediglich darin, dass Delta_Remove_Ca_Ready_For_Milk_NoSize verwendet wird, wenn das Feature Size nicht ausgewählt ist, und vor Delta_Cappuccino benutzt werden muss, während Delta_Remove_Ca_Ready_For_Milk_Size bei selektiertem Feature Size Anwendung findet und zuvor Delta_Cappuccino_Size angewendet werden muss.

Durch die Erweiterung des Verkaufsautomaten um das Angebot verschiedener Milchsorten werden 42 neue Feature-Konfigurationen und somit auch Produktvarianten hinzugefügt. Die Gesamtanzahl an gültigen Konfigurationen beträgt daher 90 Feature-Konfigurationen. Somit existieren auch 90 unterschiedliche Produktvarianten.

4.2. Scheibenwischanlage

In diesem Abschnitt werden Evolutionsszenarien für die Scheibenwischanlage beschrieben. Es ergeben sich insgesamt fünf Szenarien, welche die Erweiterung der Anlage um das Erkennen mittleren Regens (*Medium Rain*) sowie das Einfügen von unterschiedlichen Intensitätsstufen beim permanenten Wischen als auch das Hinzufügen einer Scheibenputzfunktion, einer Überprüfung des Scheibenreinigerfüllstands vor Nutzung der Scheibenputzfunktion und einer Prüfung der Scheibentemperatur vor jeder Wischerbewegung einschließen.

4.2.1. Erkennung von mittlerem Regen

Dieses Szenario beschreibt das Ergänzen der erkannten Regenstärken um die Stärke *Medium Rain*.

Beschreibung

Ausgangssituation: Die Scheibenwischanlage besteht aus einem Sensor und einem Wischer, die je in den Qualitätsstufen hoch oder niedrig angeboten werden und beliebig miteinander kombiniert werden können. Der hochwertige Sensor erkennt die Regenstärken *leicht* und *stark*, der minderwertige Sensor erkennt nur Regen allgemein. Zusätzlich besteht zu jeder Kombination die Möglichkeit, eine permanente Wischfunktion zu wählen.

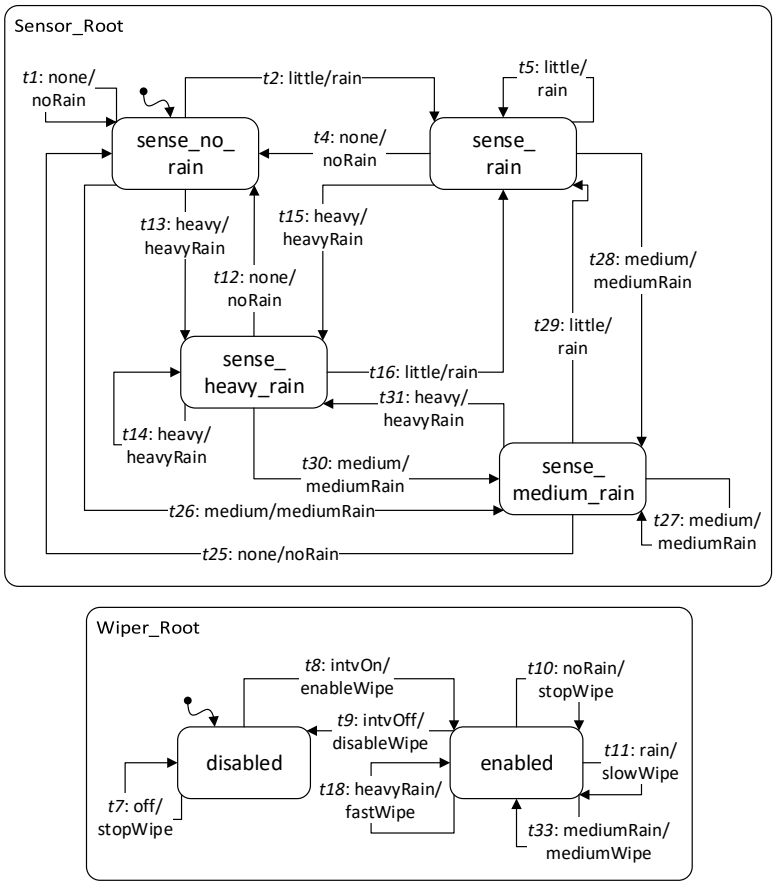

Abbildung 4.26.: Produktmodell für eine Scheibenwischanlage mit Erkennung von mittlerem Regen

Szenario: Die Kunden sind unzufrieden mit ihrer hochwertigen Sensor-Wischer-Kombination. Bei leichtem Regen funktioniere diese noch blendend, doch sobald der Regen etwas stärker werde, brauche die Anlage zu lange, um auf schnelles Wischen umzuschalten. In dieser Phase sehe man durch den Regen kaum etwas, was schon zu gefährlichen Situationen geführt habe. Sobald der Regen allerdings doch einmal eine Stärke erreiche, die das schnelle Wischen auslöst, sei dieses unverhältnismäßig schnell und störe mehr, als dass es helfe. Um den Forderungen der Kunden gerecht zu werden, beschließt der Scheibenwischanlagenhersteller daher, für den hochwertigen Sensor und den hochwertigen Wischer die Behandlung einer mittleren Stufe serienmäßig einzubauen.

Umsetzung: Da es sich bei diesem Szenario um eine interne Veränderung des Verhaltens der Sensoren und Wischer handelt, ändert sich das Feature-Diagramm der Scheibenwischanlage nicht. Dieses stimmt weiterhin mit dem Feature-Diagramm in Abbildung 3.4 in Kapitel 3.2 überein.

Das in Abbildung 4.26 abgebildete Produktmodell zeigt eine Kombination aus einem hochwertigen Wischer und einem hochwertigen Sensor. Der Sensor erkennt nun auch mittleren Regen und enthält die nötigen Übergänge, um zwischen der Erkennung von keinem, leichtem oder schwerem und mittlerem Regen zu wechseln. Der hochwertige Wischer reagiert auf die Nachricht über mittleren Regen nun mit einem mittelschnellen Wischen.

Modellierung

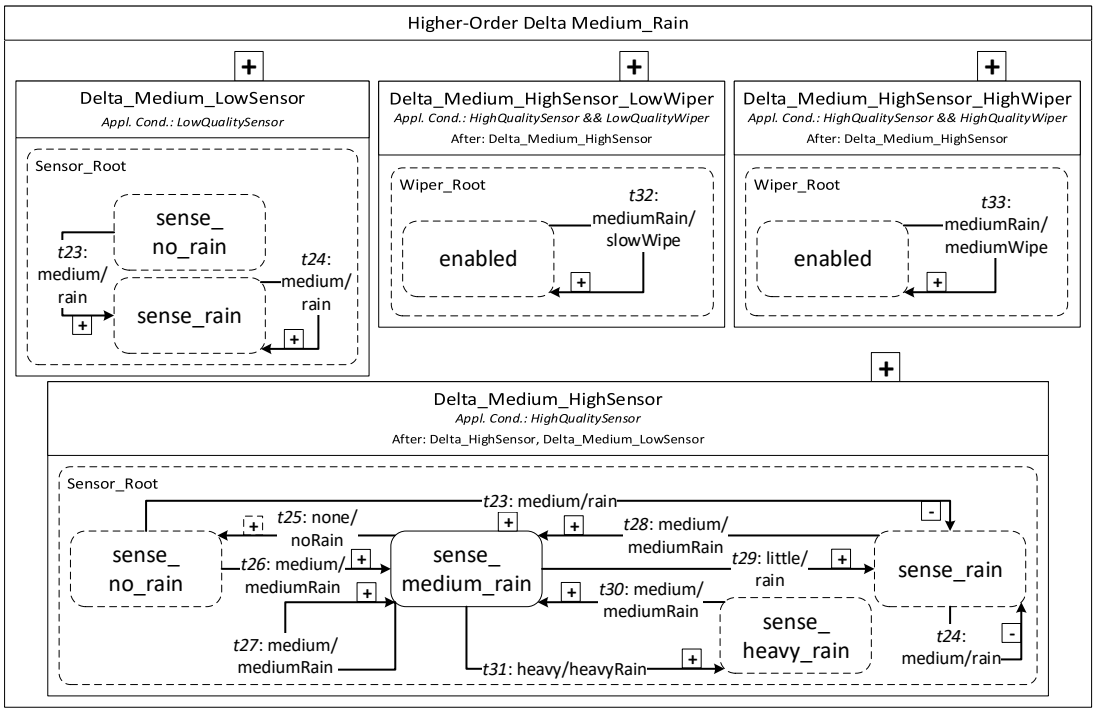

Abbildung 4.27.: Higher-Order Delta zum Hinzufügen der Erkennung von mittlerem Regen

Zur Umsetzung der Erkennung von mittlerem Regen im Delta-Modell werden die vier durch das Higher-Order Delta `Medium_Rain` neu hinzugefügten Deltas benötigt. `Delta_Medium_LowSensor` wird bei einem minderwertigen Sensor verwendet und übersetzt durch die eingefügten Transitionen `t23` und `t24` die Erkennung von mittlerem Regen lediglich in die Nachricht `rain`.

Durch `Delta_Medium_HighSensor` wird der Zustand `sense_medium_rain` ergänzt. Des Weiteren werden die Transitionen `t25` und `t26` zum Wechseln zwischen keinem und mittlerem Regen, `t28` und `t29` als Übergänge zwischen leichtem und mittlerem Regen und `t30` und `t31` für die Umstellung zwischen starkem und mittlerem Regen und `t27` zum Beibehalten von mittlerem Regen hinzugefügt. Die Transitionen `t23` und `t24` werden entfernt.

`Delta_Medium_HighSensor_LowWiper` fügt zum Anpassen des minderwertigen Wischers an die Signale des hochwertigen Sensors die Transition `t32` ein. Durch diese wird auf mittleren Regen mit einem langsamen Wischen reagiert. `Delta_Medium_HighSensor_HighWiper` ist das entsprechende Pendant für den hochwertigen Wischer und ergänzt daher die Transition `t33`, welche bei mittlerem Regen ein mittelschnelles Wischen auslöst.

Durch dieses Evolutionsszenario wird die Menge an möglichen Feature-Konfigurationen nicht verändert, allerdings werden sämtliche, vorhandene Produktvarianten modifiziert. Somit ergibt sich weiterhin eine Gesamtanzahl von acht Feature-Konfigurationen und acht Produktvarianten für die Scheibenwischanlage.

4.2.2. Intensitäten für permanentes Wischen

In diesem Szenario wird das permanente Wischen um drei Intensitätsstufen erweitert.

Beschreibung

Ausgangssituation: Eine Scheibenwischanlage kann beliebig aus einem hoch- oder minderwertigen Wischer und einem hoch- oder minderwertigen Sensor kombiniert werden. Zu jeder Kombination lässt sich auf Wunsch eine Funktion für permanentes Wischen auswählen.

Szenario: Aufgrund sich häufender Beschwerden, dass das permanente Wischen bei schwachem Regen zu stark und bei starkem Regen zu schwach sei, entscheiden sich die Hersteller dazu, für die permanente Wischfunktion optional Intensitätsstufen anzubieten. Es soll drei Stufen geben, die der Kunde manuell selbst einstellen kann.

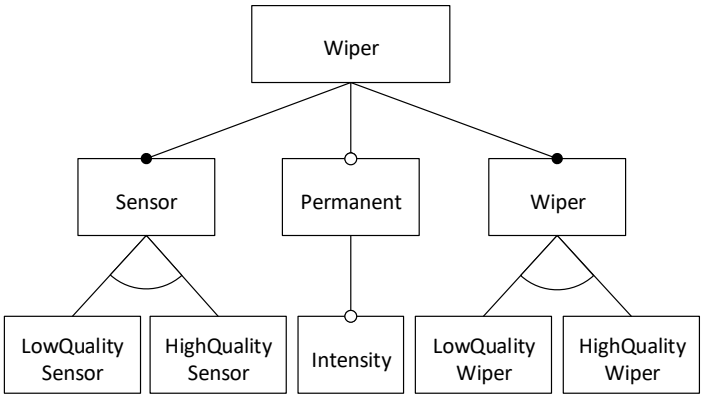

Abbildung 4.28.: Feature-Diagramm der Scheibenwischanlage mit Intensitäten für permanentes Wischen

Umsetzung: Das Feature-Modell zu diesem Szenario ist in Abbildung 4.28 dargestellt. Dort ist zu sehen, dass dem Feature `Permanent` das optionale Kind-Feature `Intensity` hinzugefügt wurde. Wird die permanente Wischfunktion selektiert, kann nun also zusätzlich gewählt werden, ob diese in einer Geschwindigkeit oder in drei zur Verfügung stehen soll.

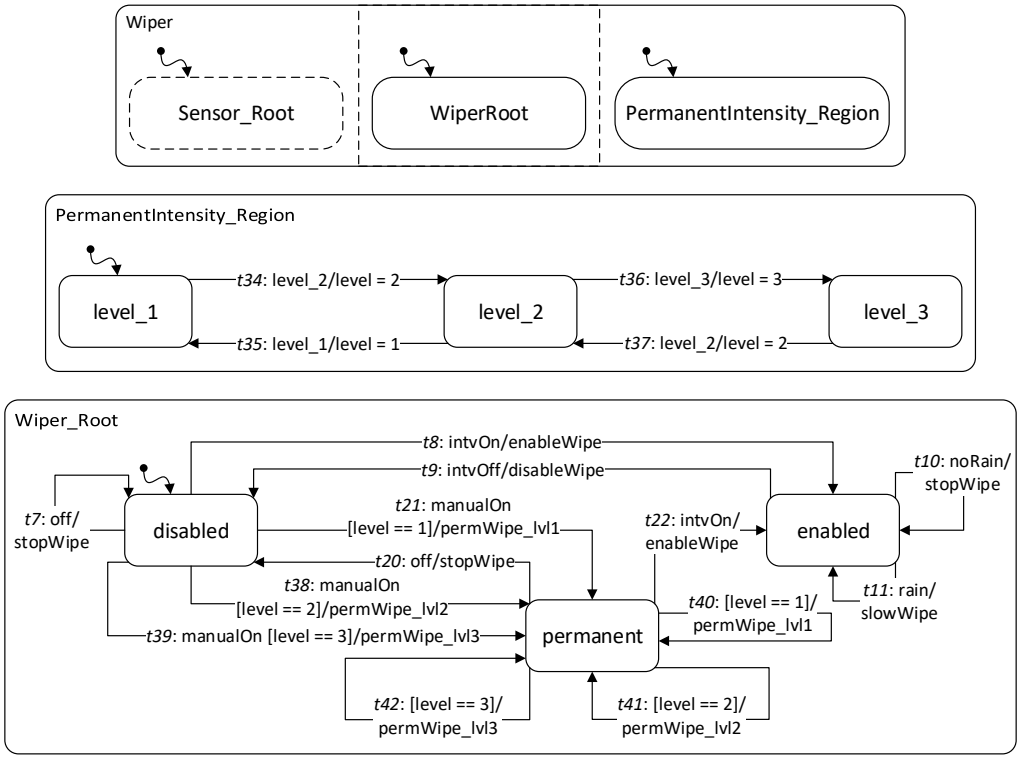

Abbildung 4.29.: Produktmodell für eine Scheibenwischanlage mit Intensitäten für permanentes Wischen

Ein Produktmodell mit den neuen Intensitäten für permanentes Wischen ist in Abbildung 4.29 abgebildet. Der State Machine wurde dabei eine neue Substate Machine hinzugefügt, die sich um die Verwaltung und Einstellung der drei verschiedenen Intensitätstufen kümmert. Im Bereich des Wischers wird nun, je nach zuvor eingestellter Intensität, bei Aktivieren des permanenten Wischens dieses automatisch auf der gewünschten Stufe ausgeführt. Zusätzlich kann während des permanenten Wischen beliebig zwischen den Intensitätsstufen gewechselt werden. Da am Sensor keine Änderungen vorgenommen werden müssen, wird dessen State Machine im Produktmodell nicht erneut abgebildet.

Modellierung

Das für diesen Evolutionsschritt benötigte Higher-Order Delta `Permanent_Intensity` ist in Abbildung 4.30 dargestellt. Es fügt die zwei neuen Deltas `Delta_Intensity` und `Delta_Permanent_Intensity` zum Delta-Modell hinzu. `Delta_Intensity` ergänzt `Wiper` um die Substate Machine `PermanentIntensity_Region`. Diese besteht aus den drei Zuständen `level_1`, `level_2` und `level_3` und den Transitionen `t34`, `t35`, `t36` und `t37`, welche für das Wechseln zwischen den Intensitätsstufen zuständig sind.

`Delta_Permanent_Intensity` modifiziert Transition `t21`, sodass diese nun bei eingestellter Stufe 1 ein permanentes Wischen der Stufe 1 auslöst. Zusätzlich werden `t38` und `t39` eingefügt, welche analog für die Stufen 2 und 3 reagieren. Darüber hinaus werden auch die Transitionen `t40`, `t41` und `t42` hinzugefügt, um auf ein Umschalten während des permanenten Wischens reagieren zu können. Dieses Delta darf erst nach `Delta_Permanent_Wiping` angewendet werden.

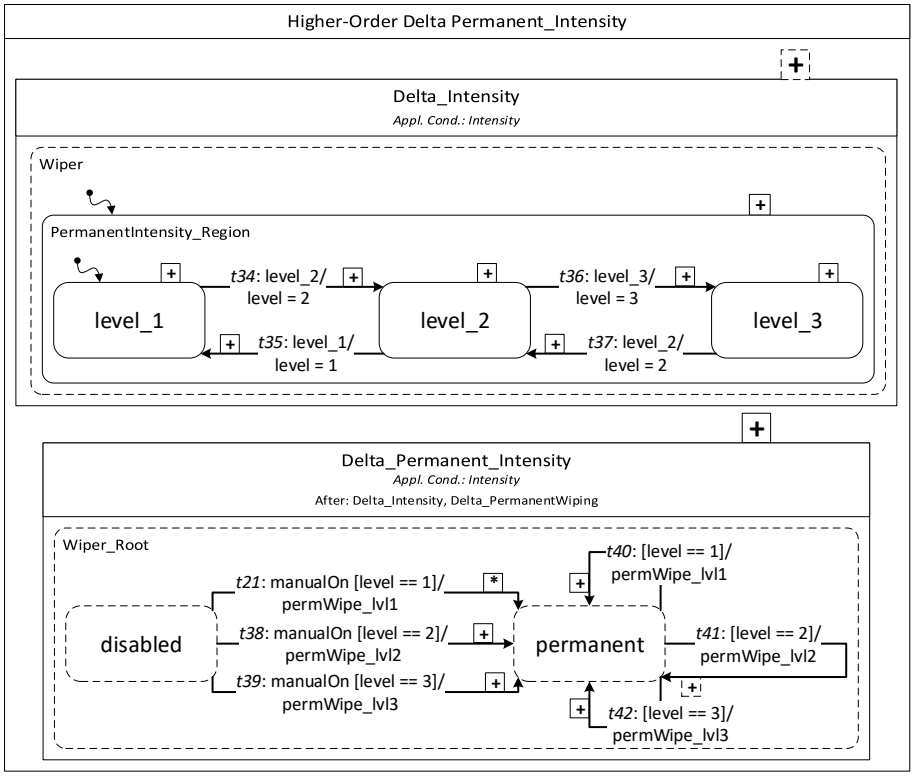

Abbildung 4.30.: Higher-Order Delta zum Hinzufügen von Intensitäten für permanentes Wischen

Das Hinzufügen der Intensitäten für permanentes Wischen ergänzt vier neue Feature-Konfigurationen und folglich auch Produktmodelle. Insgesamt ergeben sich demnach je zwölf Konfigurationen und Varianten.

4.2.3. Scheibenputzfunktion

Dieses Szenario fügt eine optionale Scheibenputzfunktion zur Produktlinie hinzu.

Beschreibung

Ausgangssituation: Die Scheibenwischanlage besteht aus einem Sensor und einem Wischer in beliebiger Kombination aus minderwertiger oder hochwertiger Qualität und kann eine permanente Wischfunktion enthalten, die wiederum drei Intensitätsstufen anbieten kann.

Szenario: Seit einiger Zeit sind die Kunden es leid, ständig an der Tankstelle anhalten zu müssen, um kleinste Verschmutzungen von ihrer Windschutzscheibe zu entfernen. Viele entscheiden sich daher neuerdings immer öfter für Scheibenwischanlage der Konkurrenz. Um dem Verlust von Kunden entgegenzuwirken, entscheidet sich der Anlagenhersteller ebenfalls eine Scheibenputzfunktion zu seiner Scheibenwischanlage anzubieten.

Umsetzung: Im Feature-Diagramm wird die Scheibenputzfunktion durch das neue, optionale Feature `Clean` repräsentiert, wie in Abbildung 4.31 dargestellt.

Die Funktion dieses Features ist es, auf Knopfdruck aus jedem möglichen Wischerzustand (automatisch, permanent oder ausgeschaltet) Scheibenreiniger zu verprühen und dann dreimal zu wischen, um danach wieder in den vorherigen Zustand zurückzufallen. Das Produktmodell in Abbil-

dung 4.32 zeigt das neue Verhalten eines minderwertigen Wischers mit permanenter Wischfunktion für dieses Feature. Der Sensor wird für dieses Szenario erneut nicht verändert und daher nicht zusätzlich dargestellt.

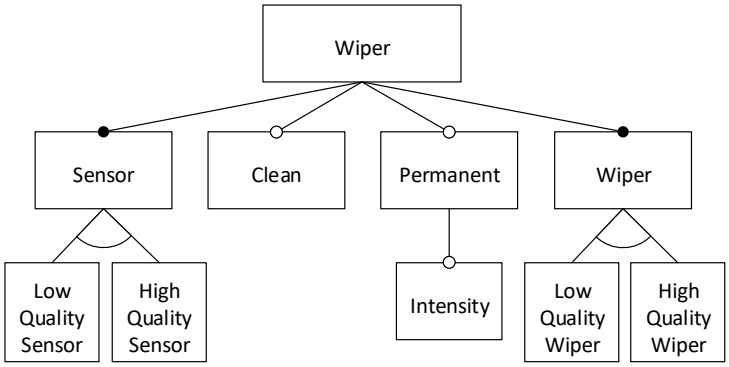

Abbildung 4.31.: Feature-Diagramm der Scheibenwischanlage mit Scheibenputzfunktion

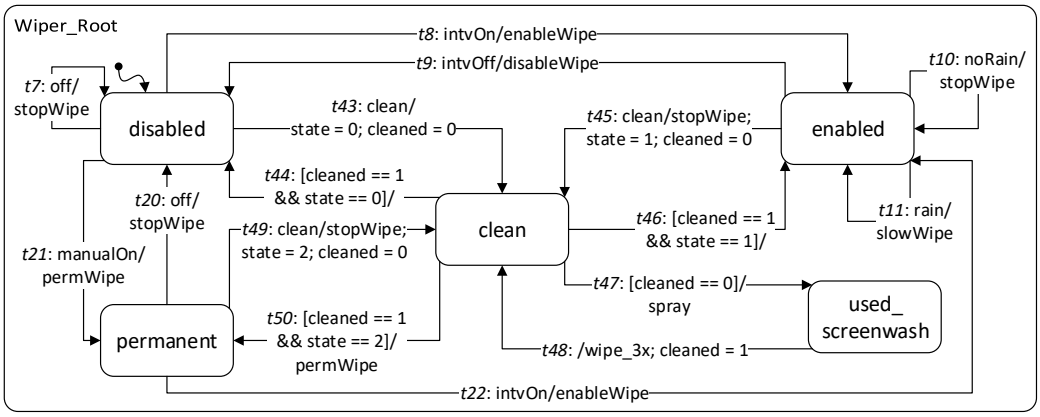

Abbildung 4.32.: Produktmodell für eine Scheibenwischanlage mit Scheibenputzfunktion

Modellierung

Damit die Scheibenputzfunktion umgesetzt werden kann, fügt das Higher-Order Delta `Clean` in Abbildung 4.33 drei neue Deltas hinzu. `Delta_Clean` ergänzt den Wischer um die Zustände `clean` und `used_screenwash` und um die Transitionen `t43` und `t45` sowie `t44` und `t46` zum Aktivieren der Putzfunktion aus den Zuständen `disabled` und `enabled` beziehungsweise Zurückkehren in diese Zustände nach erfolgtem Putzen. Ebenfalls werden die Transitionen `t47` und `t48` zum Sprühen von Scheibenreiniger und Aktivieren von dreimaligem Wischen hinzugefügt.

`Delta_Clean_Permanent` findet Anwendung, wenn zusätzlich zum Feature `Clean` auch das Feature `Permanent` gewählt wurde. In diesem Fall werden die Transitionen `t49` und `t50` zum Starten der Putzfunktion und zum erneuten Aktivieren des permanenten Wischens nach dem Putzen eingefügt. Vor diesem Delta müssen die Deltas `Delta_Clean` und `Delta_PermanentWiping` verwendet worden sein.

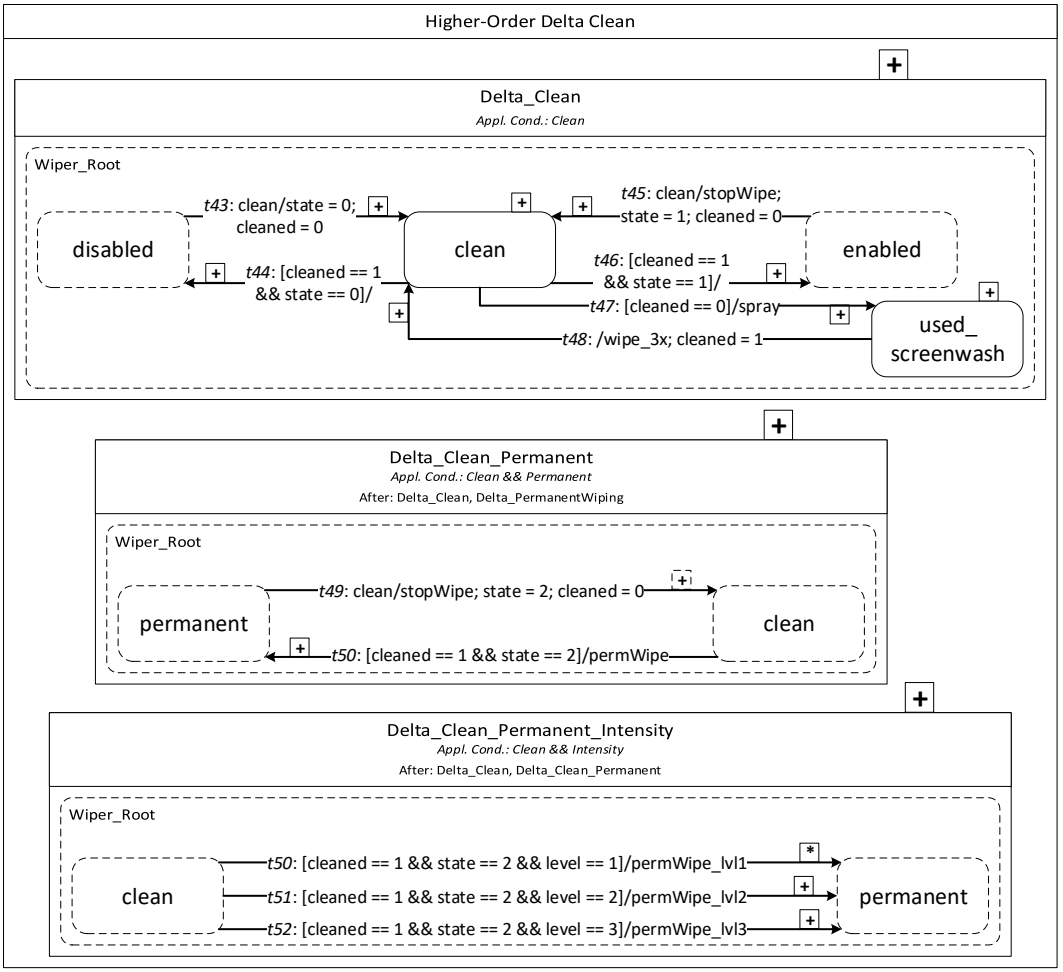

Abbildung 4.33.: Higher-Order Delta zum Hinzufügen einer Scheibenputzfunktion

Sollte darüber hinaus auch das Feature `Intensity` selektiert sein, wird `Delta_Permanent_Intensity` angewendet, um Transition `t50` zu modifizieren, sodass diese nun, bei entsprechend eingestellter Stufe, die Stufe 1 des permanenten Wischens aufruft. Ebenfalls werden `t51` und `t52` ergänzt, damit selbiges Vehalten für die Stufen 2 und 3 abgedeckt ist. `Delta_Permanent_Intensity` darf nur nach `Delta_Clean_Permanent` benutzt werden.

Dieses Szenario fügt der Scheibenwischanlage je zwölf Feature-Konfigurationen und Produktvarianten hinzu. Damit beläuft sich die Gesamtanzahl beider auf 24.

4.2.4. Überprüfung des Scheibenreinigerfüllstands

In diesem Szenario wird die automatische Überprüfung des Scheibenreinigerfüllstands eingefügt.

Beschreibung

Ausgangssituation: Die Scheibenwischanlage setzt sich aus einer beliebigen Kombination von je einem minder- oder hochwertigen Sensor und Wischer zusammen und kann zusätzlich eine permanente Wischfunktion aufweisen, die wahlweise stufenverstellbar sein kann. Außerdem kann auf Wunsch eine Scheibenputzfuntion ausgewählt werden.

Szenario: Immer häufiger werden die Gummi-Profile an den Scheibenwischern dadurch beschädigt, dass die Scheibenputzfunktion verwendet wird, obwohl nicht mehr genug Scheibenreiniger vorhanden ist, und somit ein ungewolltes Wischen auf trockener Scheibe stattfindet. Um diesem Effekt entgegenzuwirken, soll ab sofort eine Funktion standardmäßig zur Scheibenputzfunktion mitgeliefert werden, die automatisch den Füllstand des Scheibenreinigers überprüft, bevor geputzt wird. Der Scheibenreiniger kann allerdings nur durch den Service wieder aufgefüllt werden.

Umsetzung: Da die neue Funktion serienmäßig zu jeder Anlage geliefert werden soll, die auch die Scheibenputzfunktion enthält, wird sie intern im Feature `Clean` umgesetzt, sodass keine Änderung am Feature-Diagramm aus Abbildung 4.31 erfolgt.

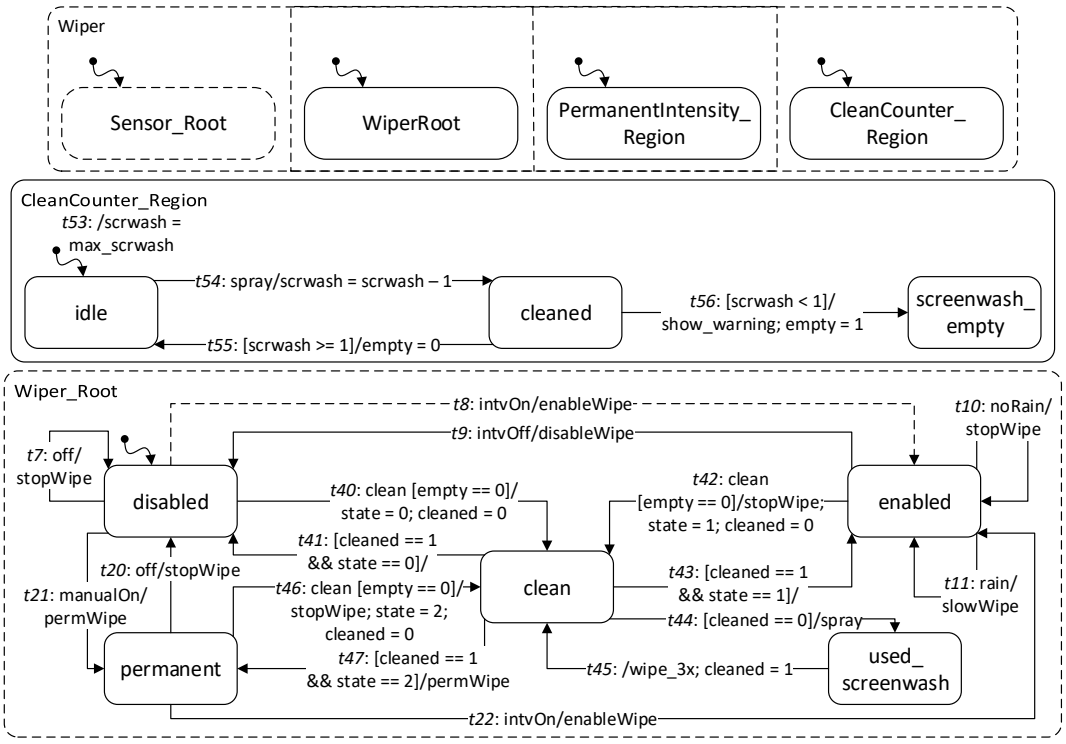

Abbildung 4.34.: Produktmodell für eine Scheibenwischanlage mit Überprüfung des Scheibenreinigerfüllstands

Abbildung 4.34 zeigt das Produktmodell einer minderwertigen Scheibenwischanlage mit permanenter Wischfunktion und überarbeiteter Scheibenputzfunktion. Um die Überprüfung des Scheibenreinigerstands umzusetzen, wurde eine neue Substate Machine hinzugefügt, die bei jedem Sprühen der Putzfunktion um eine Scheibenreinigereinheit runterzählt und bei leerem Stand eine Warnung anzeigt. Zusätzlich wird bei jedem Versuch, die Putzfunktion zu aktiveren, verglichen, ob noch genug Scheibenreiniger vorhanden ist. Sollte dies nicht der Fall sein, wird bereits der Übergang in den Zustand `clean` geblockt. Auch bei diesem Szenario erfolgen keine Änderungen am Sensor, sodass dessen Substate Machine nicht gesondert gezeigt wird.

Modellierung

Das Higher-Order Delta `Clean_Counter`, abgebildet in Abbildung 4.35, fügt ein neues Delta hinzu und modifiziert zwei bestehende Deltas. Das neue Delta `Delta_Clean_Counter` fügt die neue Substate Machine `CleanCounter_Region` zur Scheibenwischanlage hinzu. Diese setzt sich zusammen aus den Zuständen `idle`, `cleaned` und `screenwash_empty` sowie den Transitionen `t53` zum Initialisieren des Füllstands, `t54` zum Zähler runtersetzen beim Sprühen, `t55` zum Zurückkehren in den Wartezustand bei ausreichend Scheibenreiniger und `t56` zum Anzeigen einer Warnung bei zu niedrigem Scheibenreinigerstand.

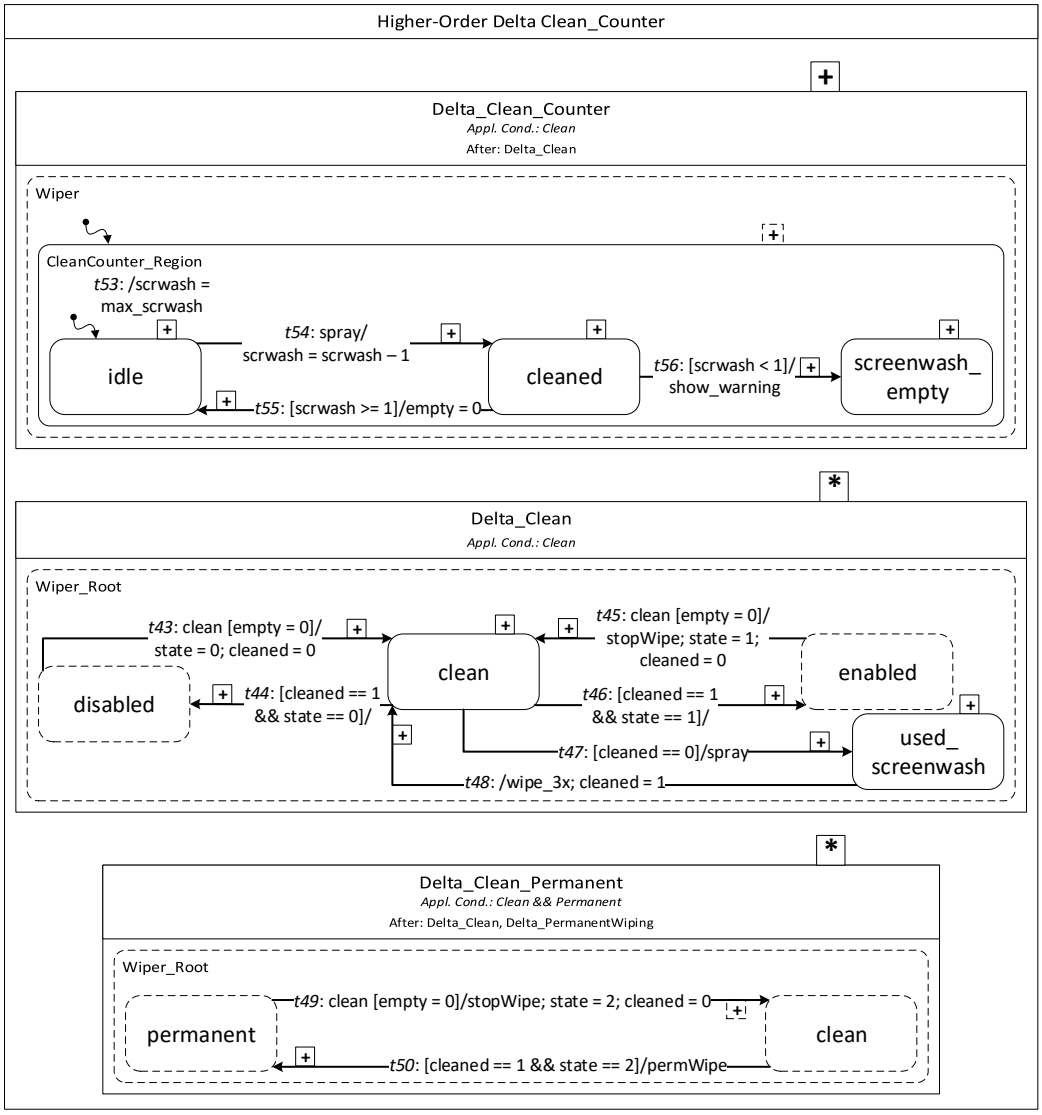

Abbildung 4.35.: Higher-Order Delta zum Hinzufügen der Überprüfung des Scheibenreinigerfüllstands

`Delta_Clean` wird modifiziert, sodass die Transitionen `t43` und `t45` nun die Bedingung enthalten, dass der Scheibenreinigerstand nicht leer sein darf. Ebenfalls wird `Delta_Clean_Permanent` modifiziert, sodass diese Bedingung auch für `t49` gilt.

Durch dieses Szenario wird die Menge an Feature-Konfigurationen nicht verändert, jedoch wird das Feature `Clean` intern erweitert. Dementsprechendend werden die zwölf Produktmodelle, welche die Scheibenputzfunktion enthalten, modifiziert. Insgesamt ergeben sich somit weiterhin je 24 Feature-Konfigurationen und Produktmodelle.

4.2.5. Frostüberprüfung

Dieses Szenario ergänzt die Überprüfung der Scheibentemperatur vor jeglicher Bewegung des Wischers.

Beschreibung

Ausgangssituation: Eine Scheibenwischanlage besteht aus einem Sensor und einem Wischer, je entweder in hoch- oder minderwertiger Ausführung, kann eine permanente Wischfunktion wahlweise mit unterschiedlichen Intensitätsstufen und eine Scheibenputzfunktion mit Scheibenreinigerfüllstandsprüfung besitzen.

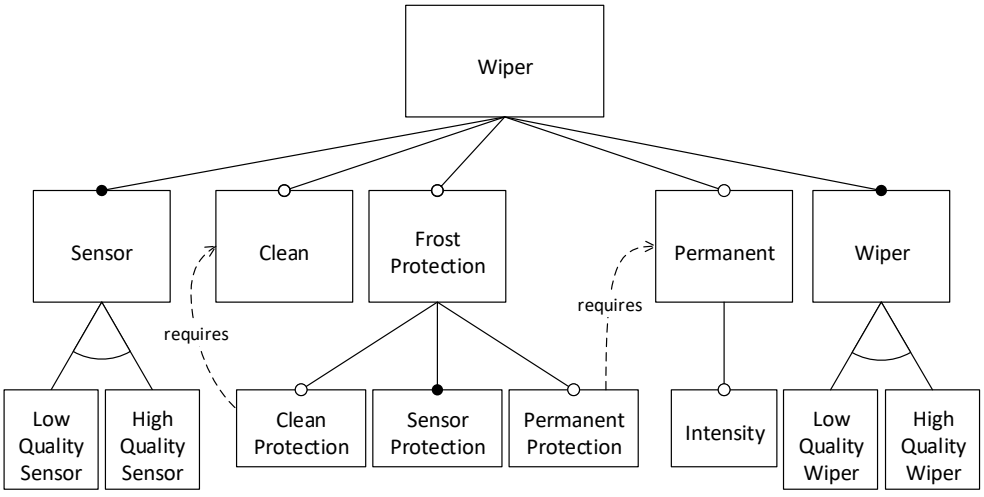

Abbildung 4.36.: Feature-Diagramm der Scheibenwischanlage mit Frostüberprüfung

Szenario: Der Winter naht und die Temperaturen erreichen in einigen Regionen schon Minusgrade. In diesen Regionen treten auch immer häufiger Schäden an den Motoren der Scheibenwischer auf, da diese durch die Temperaturen bei gerade erst gestartetem Motor noch an der Scheibe festgefroren sind, wenn schon das automatische Wischen einsetzt oder das permanente Wischen beziehungsweise die Scheibenputzfunktion aktiviert werden. Zur Verhinderung solcher Schäden soll zukünftig die Möglichkeit bestehen, eine Frostüberprüfung zu wählen.

Umsetzung: Das Feature-Diagramm wird um das neue Feature `Frost Protection` erweitert, dargestellt in Abbildung 4.36. `Frost Protection` besitzt die drei Kind-Features `Clean Protection`, `Sensor Protection` und `Permanent Protection`. `Sensor Protection` ist ein Mandatory-Feature und muss somit gewählt werden, sobald `Frost Protection` gewählt wird. `Clean Protection` und `Permanent Protection` sind Optional-Features. Wenn `Clean Protection` gewählt wird, muss auch `Clean` gewählt werden, angezeigt durch die `requires`-Relation. Genauso verhält es sich bei den Features `Permanent Protection` und `Permanent`.

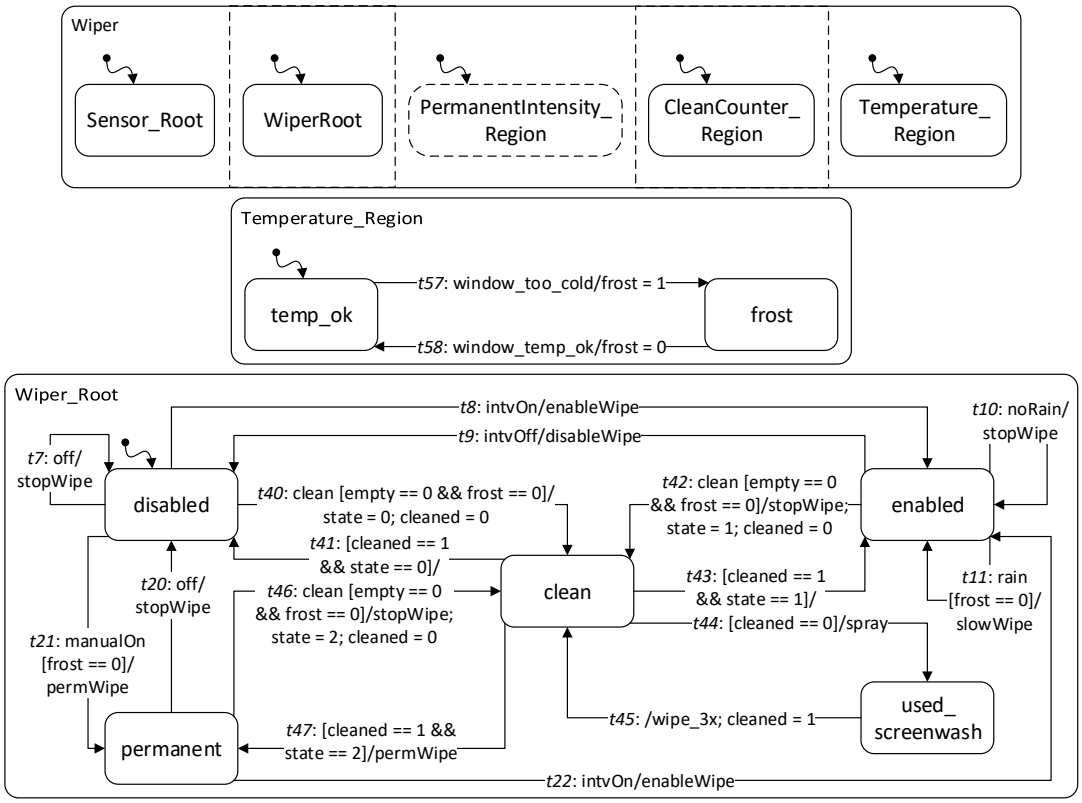

Abbildung 4.37.: Produktmodell für eine Scheibenwischanlage mit Frostüberprüfung

Das Produktmodell für einen minderwertigen Wischer mit permanenter Wischfunktion, Scheibenputzfunktion und der neuen Frostüberprüfung wird in Abbildung 4.37 gezeigt. Für die Frostüberprüfung wurde eine Substate Machine hinzugefügt, welche bei zu kalter Temperatur der Frontscheibe in einen Zustand wechselt, der angibt, dass Frost herrscht und somit der Wischer an der Scheibe festgefroren sein könnte. Der Wischer nutzt diese Information und prüft vor jedem wischauslösenden Befehl (automatisches Wischen, permanentes Wischen oder Scheibenputzfunktion), ob Frost vorherrscht und verhindert in diesem Fall die Bewegung.

Modellierung

Für Frostüberprüfung müssen elf neue Deltas durch das Higher-Order Delta `Frost_Protection` zum Delta-Modell hinzugefügt werden. Dieses ist in den Abbildungen 4.38 und 4.39 dargestellt. `Delta_Temperature` fügt die neue Substate Machine `Temperature_Region` samt Zuständen `temp_ok` und `frost` und die Transitionen `t57` bei zu kalter Fenstertemperatur und `t58` bei positiver Fenstertemperatur hinzu. `Delta_FP` modifiziert für jede Sensor-Wischer-Kombiniation außer `Low Quality Sensor` und `High Quality Wiper` die Transition `t11`, sodass diese nur bei keinem Frost das Wischen auslöst. `Delta_HighSensor_LowWiper_FP`, `Delta_HighSensor_HighWiper_FP`, `Delta_LowSensor_HighWiper_FP`, `Delta_Medium_HighSensor_LowWiper_FP` und `Delta_Medium_HighSensor_HighWiper_FP` nehmen bei den Transitionen `t17`, `t18`, `t19`, `t32` und `t33` dieselbe Modifikation vor. Diese Deltas dürfen jeweils nur nach den entsprechenden Deltas angewendet werden, welche die zu modifizierenden Transitionen einfügen.

Delta_Clean_FP passt für das gewählte Feature Clean Protection die Bedingungen der Transitionen t43 und t45 an die Frostüberprüfung an und darf nicht vor Delta_Clean verwendet werden. Delta_Clean_Permanent_FP findet Anwendung, wenn sowohl Clean Protection als auch Permanent selektiert sind, und modifiziert Transition t49, sodass auch deren Übergangsbedingung die Temperaturüberprüfung enthält. Vor Benutzung dieses Deltas muss Delta_Clean_Permanent angewendet werden.

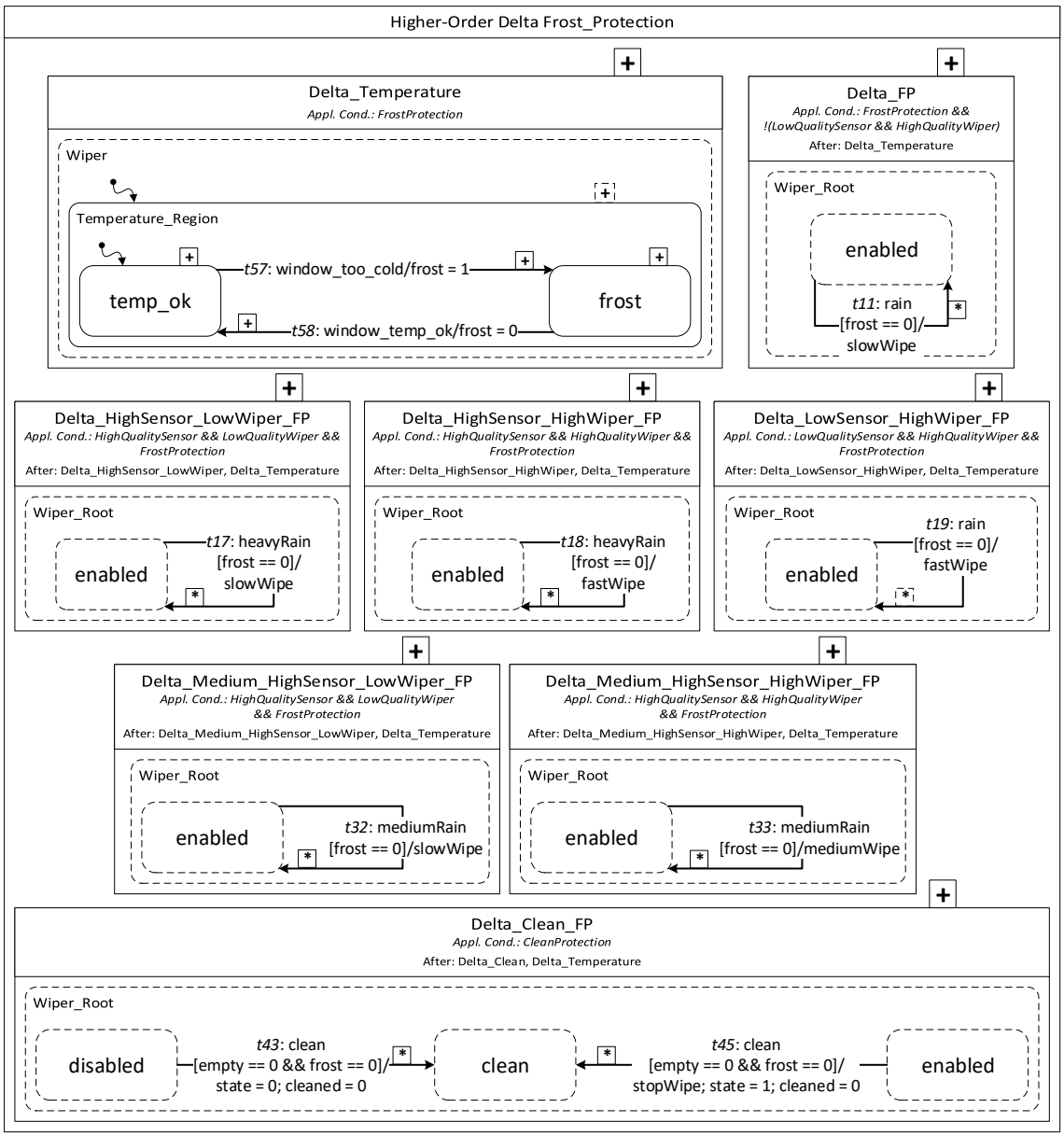

Abbildung 4.38.: Erster Teil des Higher-Order Deltas zum Hinzufügen der Frostüberprüfung

Delta_PermanentWiping_FP modifiziert für Permanent Protection auf die selbe Weise Transition t21 bei nicht gewähltem Feature Intensity und Delta_Permanent_Intensity_FP modifiziert t21 bei gewähltem Feature Intensity. Zusätzlich modifiziert Delta_Permanent_Intensity_

FP auch die Transitionen t38 und t39, sodass sie die Übergangsbedingung enthalten. Delta_PermanentWiping_FP und Delta_Permanent_Intensity_FP dürfen erst nach Delta_PermanentWiping beziehungsweise nach Delta_Permanent_Intensity benutzt werden.

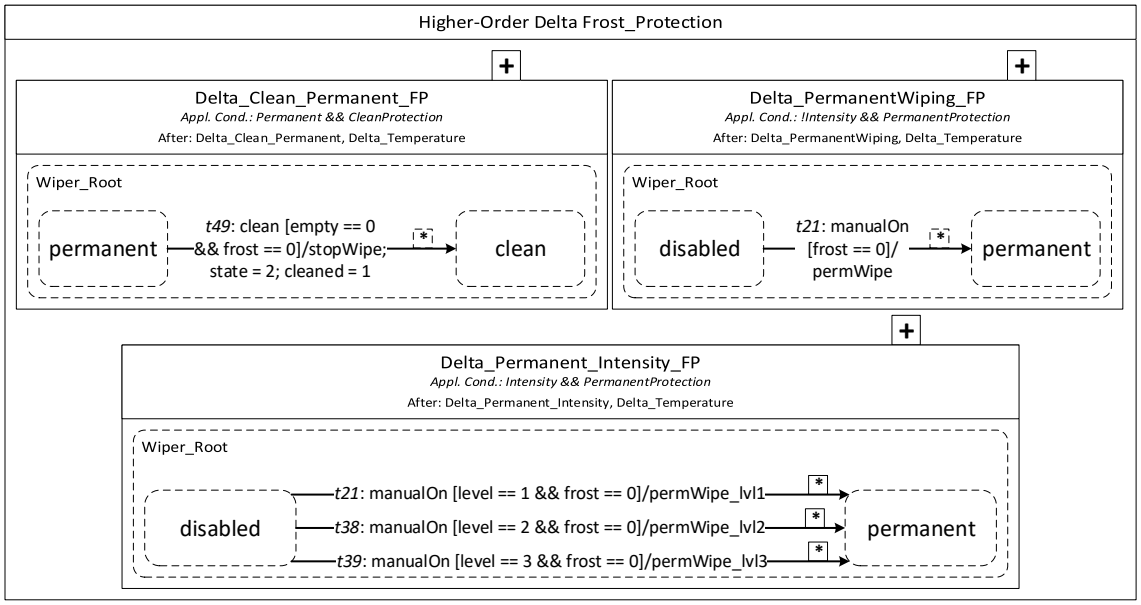

Abbildung 4.39.: Zweiter Teil des Higher-Order Deltas zum Hinzufügen der Frostüberprüfung

Das Hinzufügen der Frostüberprüfung ergänzt 60 neue Feature-Konfigurationen und somit auch 60 neue Produktmodelle. Die Gesamtanzahl der gültigen Konfigurationen und Varianten für die Scheibenwischanlage beträgt somit je 84 Feature-Konfigurationen und Produktvarianten.

4.3. Minenpumpanlage

In diesem Abschnitt werden Evolutionsszenarien für die Minenpumpanlage beschrieben. Es werden drei Szenarien vorgestellt, welche das Hinzufügen einer Methanabsaugung, das Ergänzen einer Luftüberprüfung und die Erweiterung um einen Luftaustauschmechanismus umfassen.

4.3.1. Methanabsaugung

In diesem Szenario wird eine automatische Methanabsabsaugung zur Produktlinie hinzugefügt.

Beschreibung

Ausgangssituation: Eine Minenpumpanlage besteht aus einer Wasserregulation zum Überwachen des Wasserstandes mit automatischer Abpumpfunktion bei Hochwasserstand und optionalen Reaktionen auf niedrigen oder mittleren Wasserstand. Sie kann zusätzlich eine Methanerkennung, welche die Pumpe abschaltet oder ein automatisches Einschalten verhindert, sobald ein Methanschwellenwert überschritten wird, und eine Kommandofunktion zum manuellen Ein- und Ausschalten der Pumpenfunktion erhalten.

Szenario: In den letzten Monaten wurden mehrere Minenschächte überflutet, in denen Minenpumpanlagen mit Methanerkennung eingesetzt wurden. Dies ist dem Umstand zuzuschreiben, dass die Pumpen ausgeschaltet wurden, sobald Methan erkannt wurde, und aufgrund der fehlenden Möglichkeit, Methan schnell und effektiv abzuführen, stieg das Wasser zu hoch, bevor die Pumpen wieder in Betrieb genommen werden konnten. Die Hersteller wollen daher zukünftig ihre Anlage erweitern und eine Möglichkeit bieten, das Methan automatisch absaugen zu lassen, um das Abschalten der Pumpe hinauszuzögern beziehungsweise das Wiedereinschalten zu beschleunigen.

Umsetzung: Die Methanabsaugung soll in jeder Anlage mit Methanerkennung enthalten sein. Aus diesem Grund wird kein neues Feature zum Feature-Modell hinzugefügt, sondern lediglich das bestehende Feature `Methane Detection` in seiner Funktion erweitert. Das Feature-Diagramm für dieses Szenario ändert sich somit im Vergleich zum Feature-Diagramm aus Kapitel 3.3 nicht.

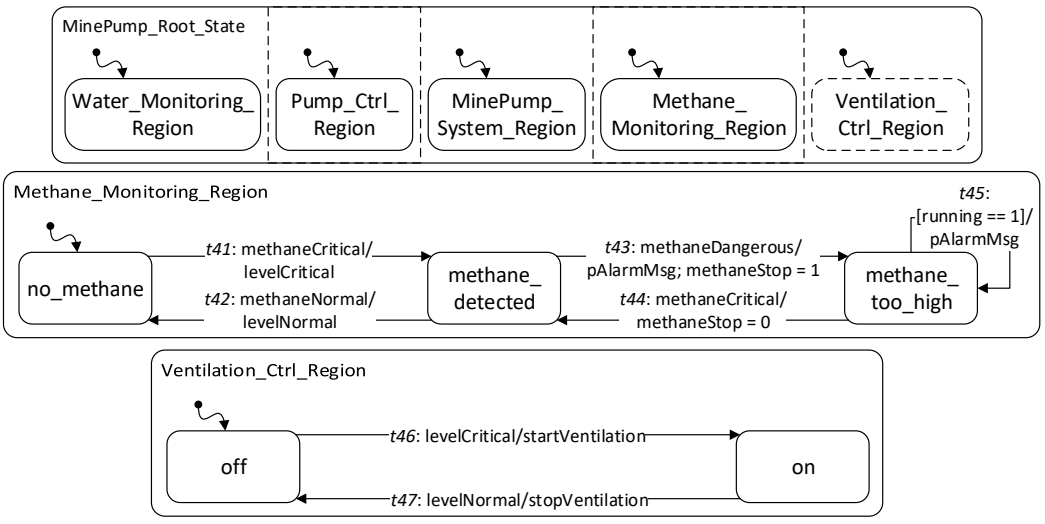

Abbildung 4.40.: Produktmodell für eine Minenpumpanlage mit Methanabsaugung

Abbildung 4.40 zeigt ein mögliches Produktmodell mit Methanabsaugung. Das Produktmodell wurde um eine Substate Machine erweitert, welche, je nach aktuellem Methangehalt in der Luft, die Absauganlage ein- oder ausschaltet. Die bisherige Methanerkennung wurde dahingehend verändert, dass sie nun in drei Stufen funktioniert. Wenn der Methangehalt auf ein kritisches Level steigt, wird bereits die Absauganlage aktiviert, die Pumpenfunktion ist aber noch möglich. Sobald der Methangehalt trotz Absaugung ein gefährliches Level erreicht, wird die Wasserpumpe ausgeschaltet beziehungsweise deren Einschaltung verhindert. Sinkt der Methangehalt wieder auf das kritische Level, kann die Pumpe erneut verwendet werden, und bei normalem Methangehalt wird auch die Absauganlage wieder abgestellt. An den anderen Substate Machines der Minenpumpanlage ergeben sich keine Änderungen, daher werden diese nicht gesondert dargestellt.

Modellierung

Da es sich bei `Methane_Extraction` um ein Mandatory-Feature handelt, wird `Delta_Add_Methane_Monitoring_Region` von dem Higher-Order Delta `Methane_Extraction` in Abbilung 4.41 modifiziert, da es in seiner alten Form nicht mehr benötigt wird. Im Vergleich zu vorher wird ein weiterer Zustand (`methane_too_high`) ergänzt und die Transitionen `t23`, `t24` und `t25` durch `t41` bis `t45` ersetzt. Transition `t41` meldet ein kritisches Methanlevel, `t43` löst den Alarm aus, wenn der Methangehalt gefährlich hoch ist, `t45` erzeugt erneut einen Alarm, wenn die Pumpe trotz vorherigem Alarm noch laufen sollte, `t44` lässt die Aktivierung der Pumpe wieder zu und `t42` zeigt an, dass der Methangehalt wieder normal ist.

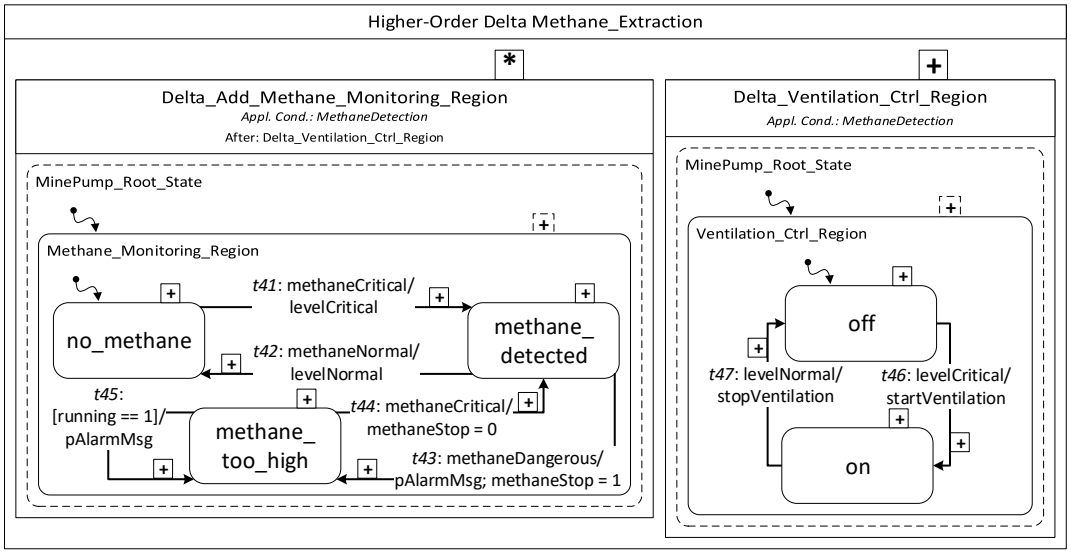

Abbildung 4.41.: Higher-Order Delta zum Hinzufügen einer Methanabsaugung

Zusätzlich fügt das HOD `Delta_Add_Ventilation_Ctrl_Region` zum Delta-Modell hinzu, welches die `Ventilation_Control_Region` ergänzt, die aus den Zuständen `on` und `off` sowie den Transitionen `t46` zum Einschalten der Absaugung bei kritischem Methangehalt und `t47` zum Ausschalten bei ungefährlichem Methangehalt besteht.

Durch dieses Szenario wird die Menge an Feature-Konfigurationen nicht verändert. Da allerdings die Funktion des Features `Methane Detection` erweitert wird, werden die acht Produktvari-

anten, welche die Methanerkennung enthalten, modifiziert. Insgesamt existieren so weiterhin je 16 Feature-Konfigurationen und Produktmodelle.

4.3.2. Luftüberprüfung

Dieses Szenario ergänzt die Minenpumpanlage um eine Überprüfung der Luftzusammensetzung.

Beschreibung

Ausgangssituation: Jede Minenpumpanlage enthält eine Wasserregulation, die den Wasserstand überwacht und bei Hochwasser automatisch beginnt Wasser abzupumpen bis eine gewisse Zeitspanne abgelaufen ist oder falls, soweit gewählt, ein Niedrigstand erkannt wird. Die Anlage kann auf Wunsch mit einer Kommandofunktion und einer Methanerkennung mit Methanabsaugung ausgestattet werden.

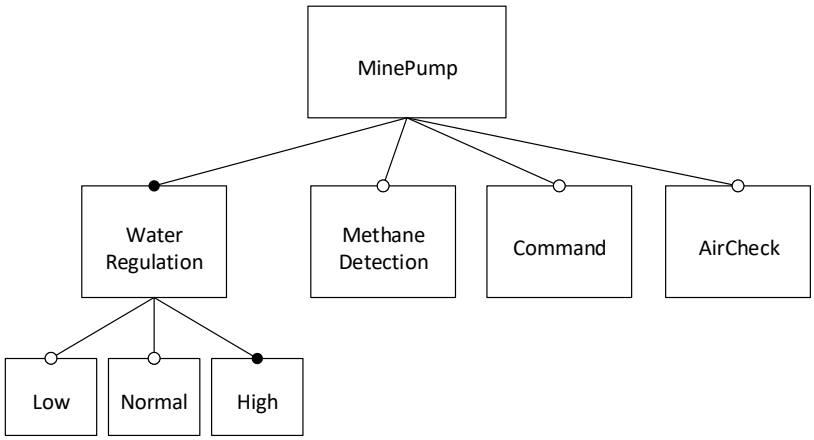

Abbildung 4.42.: Feature-Diagramm der Minenpumpanlage mit Luftüberprüfung

Szenario: Mehrere Minenarbeiter wurden mit Kohlenmonoxidvergiftungen oder Sauerstoffmangelerscheinungen ins Krankenhaus eingeliefert, nachdem die Pumpanlagen für mehrere Stunden am Stück gelaufen waren. Durch die beengten Schächte, die schlechte Luftzirkulation und die durch den Verbrennungsmotor der Pumpe ausgestoßenen Abgase wurde der Sauerstoff verdrängt und Kohlenmonoxid sammelte sich in den Schächten. Um solche Gefahren zu verhindern, sollen die Minenpumpanlagen ab sofort mit einer Luftüberprüfung ausgestattet werden, welche zu niedrigen Sauerstoffgehalt beziehungsweise zu hohen Kohlenmonoxidgehalt erkennt, eine Warnung ausgibt und die Pumpe abschaltet.

Umsetzung: Zur Umsetzung dieser Funktion wird dem Feature-Modell in Abbildung 4.42 das Optional-Feature `Air Check` hinzugefügt.

Abbildung 4.43 zeigt das Produktmodell für die Feature-Konfiguration `Water Regulation` inklusive `High`, `Methane Detection` mit `Methane Extraction` und `Air Check`. Dem Modell wurde hierbei eine neue Substate Machine für die Überwachung der Luftqualität hinzugefügt, welche einen Alarm auslöst und eine Verwendung der Wasserpumpe verhindert, wenn die Luft zu sehr verschmutzt ist beziehungsweise nicht genug Sauerstoff enthält. Zusätzlich wurde die `Methane_Level_Control_Region` durch die `Alarm_Control_Region` ersetzt, damit diese nun sowohl auf die Alarme ausgelöst durch Methan als auch Luftverschmutung reagieren kann. Die `Water_Level_Control_`

Region wird ebenfalls angepasst, sodass die Pumpe trotz hohem Wasserstand nicht nur bei zu hohem Methangehalt, sondern auch bei verschmutzter Luft nicht mehr gestartet werden kann.

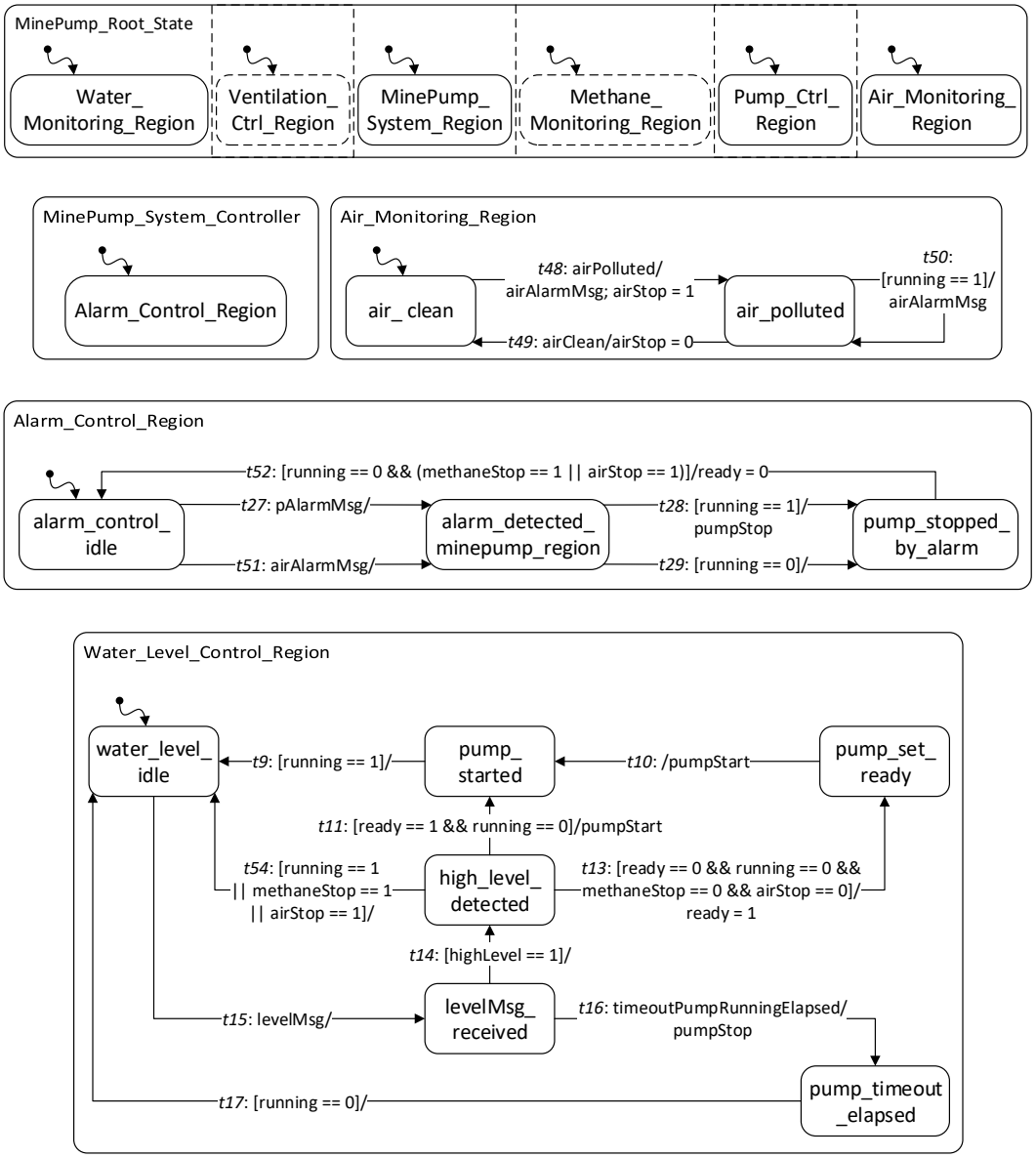

Abbildung 4.43.: Produktmodell für eine Minenpumpanlage mit Luftüberprüfung

Modellierung

Die Abbildungen 4.44, 4.45, 4.46 und 4.47 zeigen die Deltas, die für diesen Evolutionsschritt durch das Higher-Order Delta `Air_Check` hinzugefügt, modifiziert und entfernt werden. `Delta_Add_Air_Monitoring_Region` fügt die `Air_Monitoring_Region` mit den Zuständen `air_clean` und `air_polluted` sowie die Transitionen `t48` zum Auslösen des Alarms und Blockieren der Pumpe bei verschmutzter Luft, `t49` zum Erlauben erneuter Pumpenaktivität bei sauberer Luft und `t50` zum erneuten Auslösen des Alarms bei fehlgeschlagenem, ersten Versuch zum Modell hinzu.

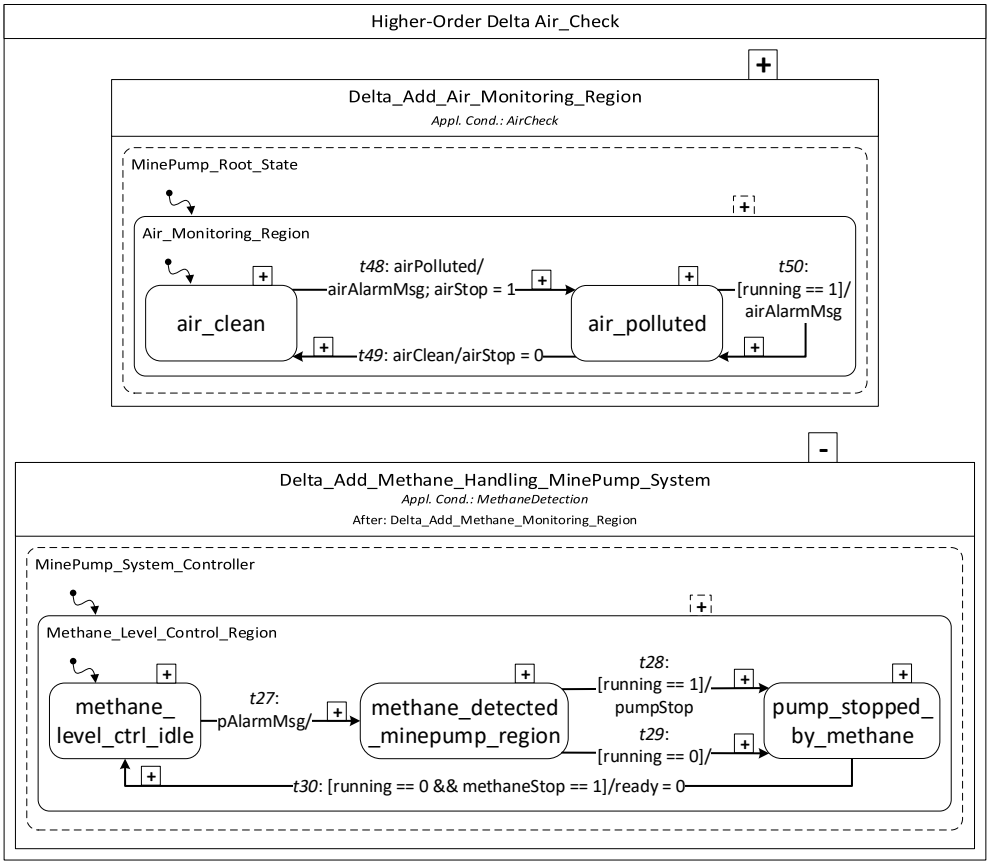

Abbildung 4.44.: Erster Teil des Higher-Order Deltas zum Hinzufügen einer Luftüberprüfung

In Abbildung 4.45 wird das Delta `Delta_Add_Methane_Handling_MinePump_System` aus dem Delta-Modell entfernt, womit die `Methane_Level_Control_Region` gänzlich verschwindet. `Delta_Alarm_Level_Control_Region` fügt stattdessen die `Alarm_Control_Region` mit den Zuständen `alarm_ctrl_idle`, `alarm_detected_minepump_region` und `pump_stopped_by_alarm` hinzu, die sowohl mit Feature `Methane Detection` als auch `Air Check` verwendet werden kann. Für `Air Check` ergänzt `Delta_Air_Alarm_Msg` in der `Alarm_Control_Region` dann die Transition `t51` zum Auslösen des Alarms bei verschmutzter Luft und `Delta_Methane_Alarm_Msg` fügt für `Methane Detection` Transition `t27` mit dem Methanalarm ein. Für beide Features werden durch `Delta_Alarm_Pump_Stop` die Transitionen `t28` zum Abschalten der laufenden Pumpe und `t29` zum Überspringen bei nicht laufender Pumpe hinzugefügt. Sollte nur `Methane Detection` gewählt sein, nicht jedoch `Air Check`, ergänzt `Delta_Methane_Handling` die Transition `t30`, um bei nicht laufender Pumpe und durch Methan ausgelöster Pumpenblockierung in den Ausgangszustand zurückzukehren. `Delta_Air_Handling` fügt zum selben Zweck, aber mit durch Luftverschmutzung ausgelöster Pumpenblockierung, `t53` für das Feature `Air Check` ein, allerdings nicht, wenn auch `Methane Detection` gewählt ist. Für den Fall, dass beide Features selektiert sind, wird stattdessen durch `Delta_Air_Methane_Handling` Transition `t52` hinzugefügt, welche sowohl bei einer Blockierung durch Methan als auch durch Luftverschmutzung in den Ausgangszustand zurückkehrt. `Delta_Air_Alarm_Msg`, `Delta_Methane_Alarm_Msg` und `Delta_Alarm_Pump_Stop` dürfen erst nach `Delta_Alarm_Level_`

Control_Region verwendet werden. Delta_Air_Alarm_Msg darf zusätzlich erst nach Delta_Add_Air_Monitoring_Region und Delta_Methane_Alarm_Msg erst nach Delta_Add_Methane_Monitoring_Region angewendet werden. Vor Delta_Methane_Handling und Delta_Air_Handlung müssen Delta_Methane_Alarm_Msg beziehungsweise Delta_Air_Alarm_Msg benutzt werden und vor Delta_Air_Methane_Handling müssen dementsprechend sowohl Delta_Methane_Alarm_Msg als auch Delta_Air_Alarm_Msg verwendet werden.

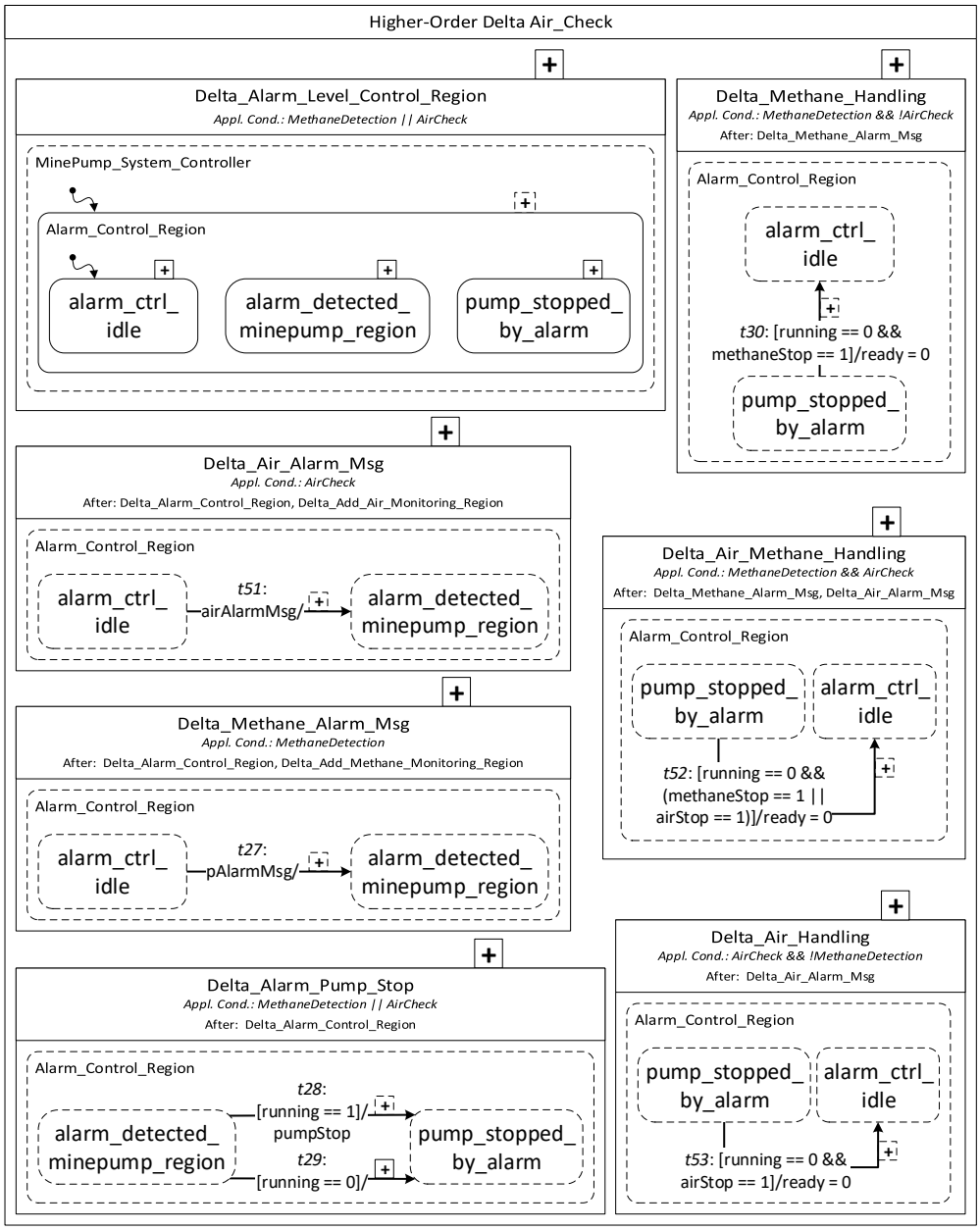

Abbildung 4.45.: Zweiter Teil des Higher-Order Deltas zum Hinzufügen einer Luftüberprüfung

Die Deltas Delta_Add_Methane_Handling_HighLevel_No_Cmd_MinePump_System, Delta_Add_Command_Handling_HighLevel_No_Methane_MinePump_System und Delta_Add_Command_And_

Methane_Handling_HighLevel_MinePump_System in Abbildung 4.46 werden durch das HOD modifiziert, sodass ihre Anwendungsbedingungen zusätzlich mit einschließen, dass das Feature Air Check nicht gewählt sein darf. Darüber hinaus werden sie im Vergleich zu vorher nicht weiter verändert.

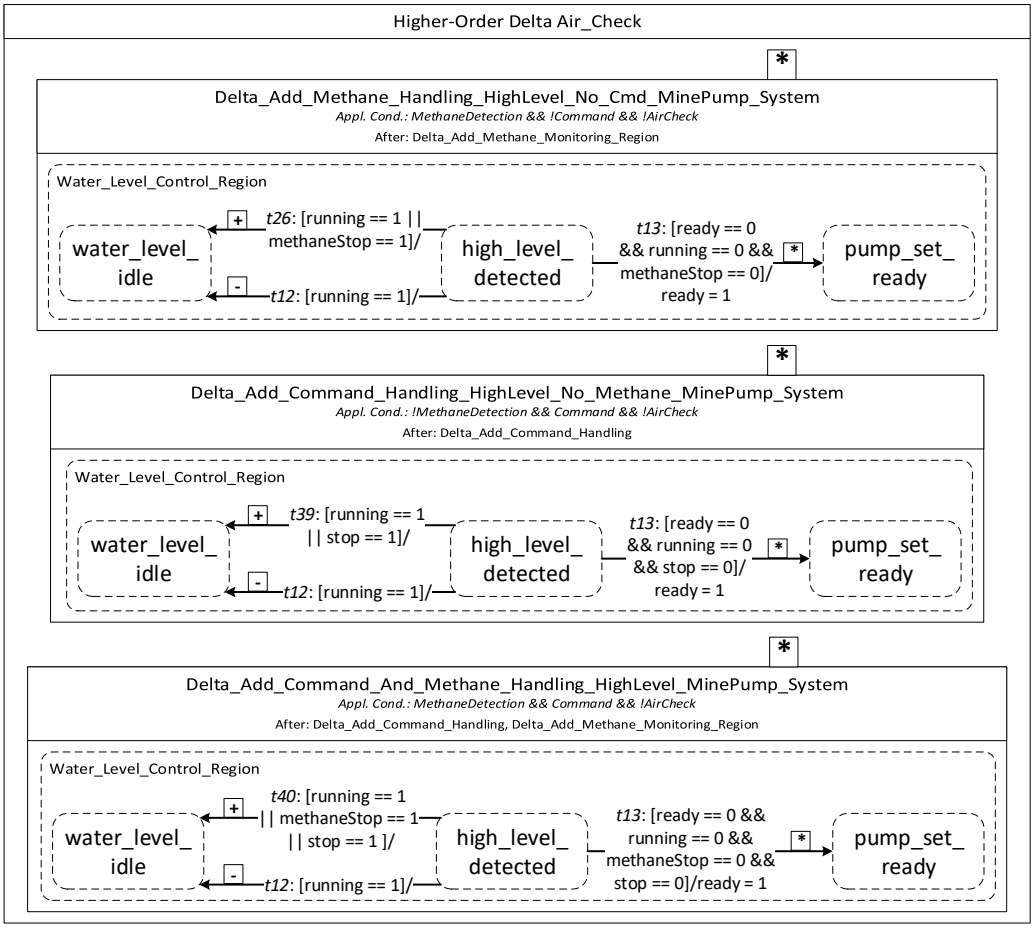

Abbildung 4.46.: Dritter Teil des Higher-Order Deltas zum Hinzufügen einer Luftüberprüfung

Zusätzlich werden in Abbildung 4.47 noch vier weitere Deltas hinzugefügt. Delta_Add_Methane_And_Air_Handling_HighLevel_No_Cmd_MinePump_System modifiziert die Transitionen t12, um trotz Hochwasserstand auch bei durch Methan oder Luftverschmutzung ausgelöstem Stop ohne Aktion in den Ausgangszustand zurückzukehren, und t13, um zusätzlich zu prüfen, ob kein Stop durch Methan oder Luftverschmutzung vorliegt, bevor die Pumpe eingeschaltet wird. Gleichermaßen modifiziert Delta_Add_Command_And_Air_Handling_HighLevel_No_Methane_MinePump_System t12 und t13 für Luftverschmutzungsstop und Kommandostop, Delta_Add_Command_Methane_And_Air_Handling_HighLevel_MinePump_System für einen Stop durch Luftverschmutzung, Methan oder Kommando und Delta_Add_Air_Handling_HighLevel_No_Command_No_Methane_MinePump_System lediglich für den Stop durch Luftverschmutzung. Vor allen vier Deltas muss zuerst Delta_Add_Air_Monitoring_Region angewendet werden. Zusätzlich darf Delta_Add_Methane_And_Air_Handling_HighLevel_No_Cmd_MinePump_System ausschließlich nach Delta_

Add_Methane_Monitoring_Region, Delta_Add_Command_And_Air_Handling_HighLevel_No_Methane_MinePump_System erst nach Delta_Add_Command_Handling und Delta_Add_Command_Methane_And_Air_Handling_HighLevel_MinePump_System nicht vor Delta_Add_Methane_Monitoring_Region und Delta_Add_Command_Handling angewendet werden.

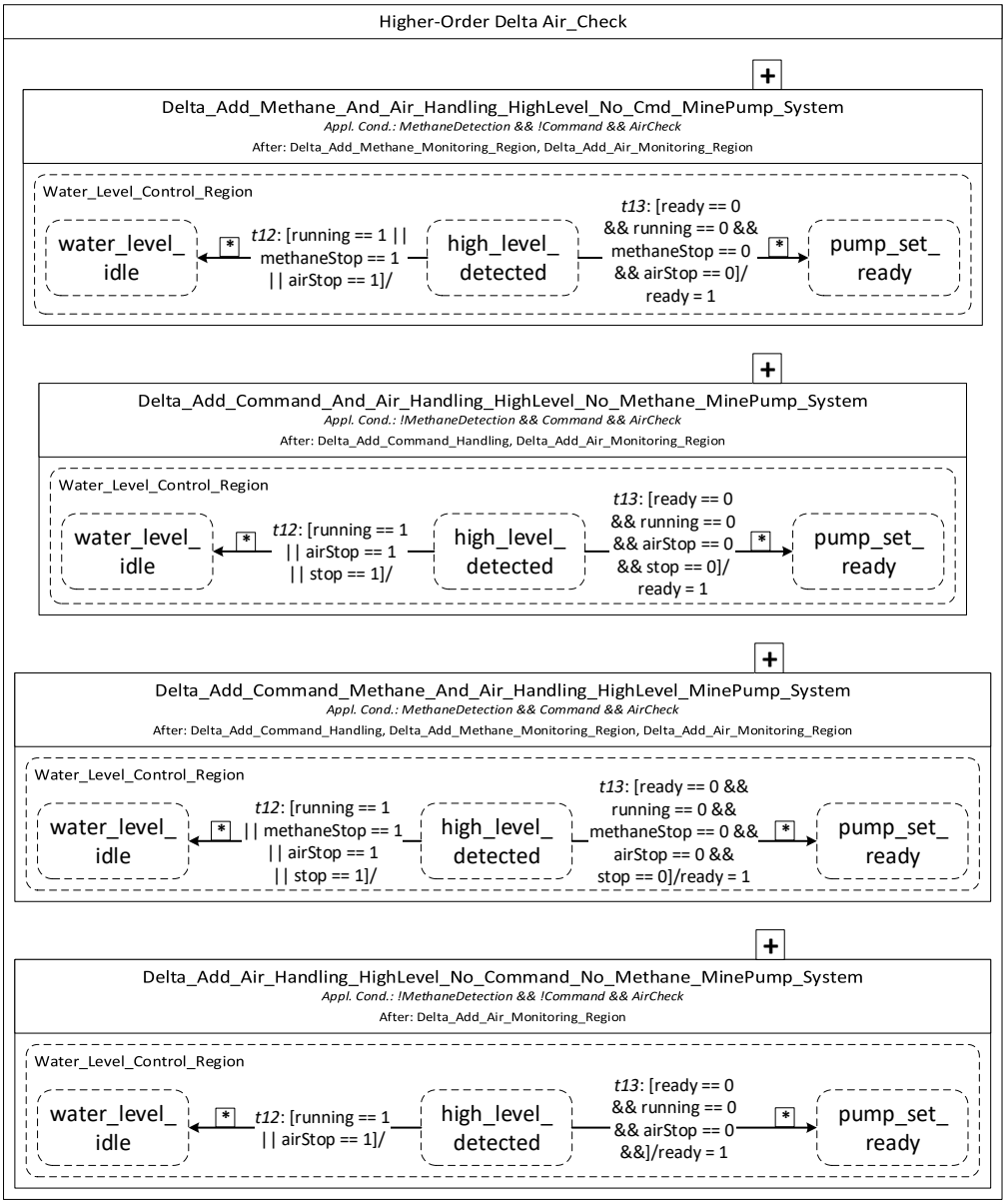

Abbildung 4.47.: Vierter Teil des Higher-Order Deltas zum Hinzufügen einer Luftüberprüfung

Dieses Szenario fügt 16 neue Feature-Konfigurationen und dementsprechend auch 16 neue Produktmodelle zur Softwareproduktlinie hinzu. Insgesamt ergeben sich somit je 32 Feature-Konfigurationen und Produktmodelle für die Minenpumpanlage.

4.3.3. Luftaustausch

Dieses Szenario fügt einen Mechanismus für automatischen Luftaustausch zur Minenpumpanlage hinzu.

Beschreibung

Ausgangssituation: Die Minenpumpanlage hat eine automatische Abpumpfunktion für Hochwasserstand und kann auf Wunsch auch Reaktionen auf Niedrig- und Normalwasserstand zeigen. Des Weiteren kann sie mit einer Methanerkennung, in welchem Fall ebenfalls eine Methanabsaugung enthalten ist, und einer Kommandofunktion ausgestattet werden. Zusätzlich besteht die Möglichkeit, eine Überprüfung der Luftqualität zu wählen, welche die Pumpe stoppt, wenn die Luftqualität zu schlecht ist.

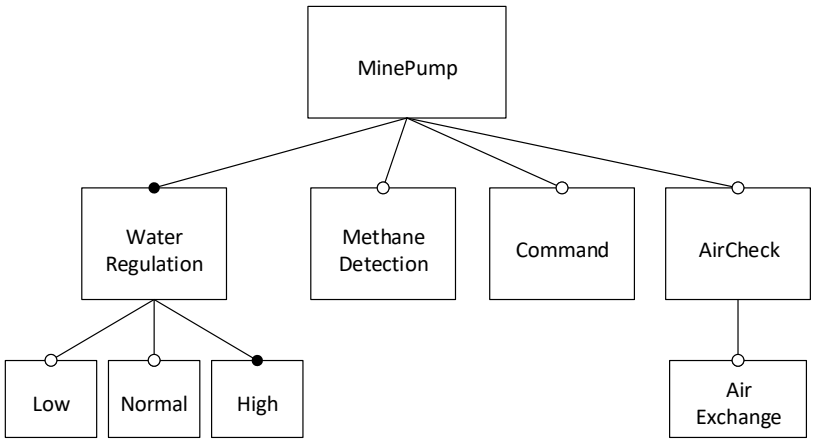

Abbildung 4.48.: Feature-Diagramm der Minenpumpanlage mit Luftaustausch

Szenario: Durch die neue Funktion der Luftüberprüfung ist es in schlecht belüfteten Minen des öfteren zu Überflutungen der Minenschächte gekommen, wenn die Funktion der Pumpe durch zu schlechte Luftqualität abgeschaltet war. Aus diesem Grund stellen die Hersteller nun auch für die Luftüberprüfung eine Luftaustauschfunktion ähnlich der Methanabsaugung zur Verfügung. Diese soll durch die gleiche Anlage geregelt sein wie auch die Methanabsaugung und je nach gewähltem Produkt eine oder beide der Funktionen umsetzen.

Umsetzung: Das Feature-Modell für dieses Szenario ist in Abbildung 4.48 zu sehen. Diesem wurde das optionale Feature `Air Exchange` als Kind-Feature von `Air Check` hinzugefügt. Wird `Air Check` selektiert, besteht nun also die Möglichkeit zu wählen, ob auch `Air Exchange` gewünscht ist.

Die für dieses Feature veränderten Teile des Produktmodells sind in Abbildung 4.49 abgebildet. Die Einteilung der Luftqualität wird hierbei, wie bereits schon bei Methan, nun in drei Stufen aufgeteilt. Bei kritischer Luftqualität wird die Luftaustauschanlage eingeschaltet und bei verschmutzter Luft wird ein Alarm ausgelöst und die Wasserpumpe ausgeschaltet beziehungsweise deren Einschaltung verhindert. Sobald die Luft vom verschmutzten wieder im kritischen Bereich ist, kann die Wasserpumpe wieder verwendet werden, während der Luftaustausch weiterhin erfolgt. Erst bei sauberer Luft wird die Anlage wieder abgeschaltet. Da es sich hier um das Modell eines Produktes mit `Methane Extraction` und `Air Exchange` handelt, wird die Luftaustauschanlage nun eingeschaltet,

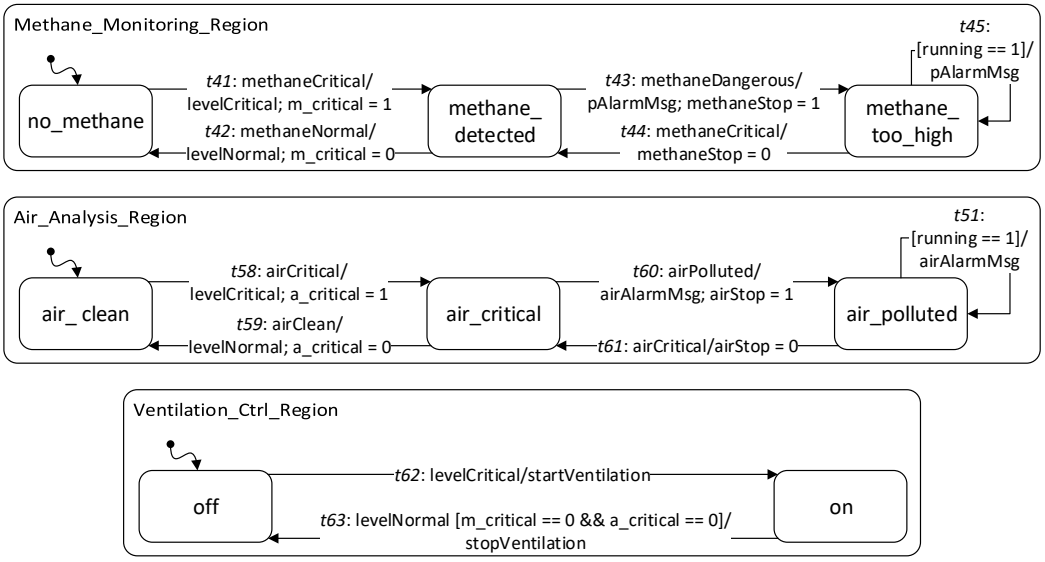

Abbildung 4.49.: Produktmodell für eine Minenpumpanlage mit Luftaustausch

wenn entweder die Luftqualität oder der Methangehalt kritisch sind. Entsprechend wird sie auch nur dann ausgeschaltet, wenn beide Werte nicht kritisch sind. Der Rest des Produktmodells bleibt unverändert und wird daher hier nicht dargestellt.

Modellierung

Für dieses Feature müssen zwei neue Deltas durch das HOD `Air_Exchange` in den Abbildungen 4.50 und 4.51 hinzugefügt und zwei Deltas modifiziert werden. Um die Überprüfung der Luftqualität anzupassen, wird durch `Delta_Air_Exchange` in `Air_Analysis_Region` der Zustand `air_critical` ergänzt und die Transitionen `t48` und `t49` entfernt. Zusätzlich werden die Transitionen `t58` für kritische und `t59` für normale Luftqualität, `t60` zum Auslösen des Alarms und des Pumpenstops bei verschmutzter Luft und `t61` zum Abschalten des Pumpenstops bei kritischer Luftqualität hinzugefügt.

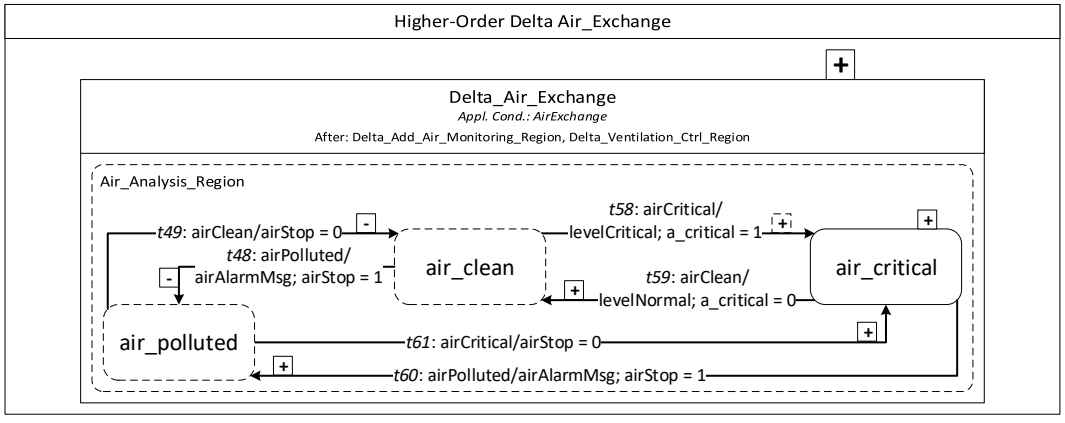

Abbildung 4.50.: Erstes Higher-Order Delta zum Hinzufügen eines Luftaustauschs

`Delta_Ventilation_Ctrl_Region` wird modifiziert, sodass dieses nun auch bei Feature `Air Exchange` angewendet werden kann. Sollten `Methane Extraction` und `Air Exchange` gleichzeitig gewählt sein, wird das neue Delta `Delta_Methane_Air_Ventilation_Ctrl` angewendet, um `t47` zu entfernen und stattdessen Transition `t63` einzufügen, welche die Lüftung nur ausschaltet, wenn sowohl der Methangehalt als auch die Luftqualität im Normalbereich liegen. Vor Verwendung dieses Deltas müssen zuerst `Delta_Ventilation_Ctrl_Region`, `Delta_Add_Methane_Monitoring_Region` und `Delta_Air_Exchange` benutzt werden. Damit die Übergangsbedingung für Transition `t63` umgesetzt werden kann, wird außerdem `Delta_Add_Methane_Monitoring_Region` modifiziert, sodass die Transitionen `t41` und `t42` nun zusätzlich in einer Variable speichern, ob der Methanwert kritisch ist.

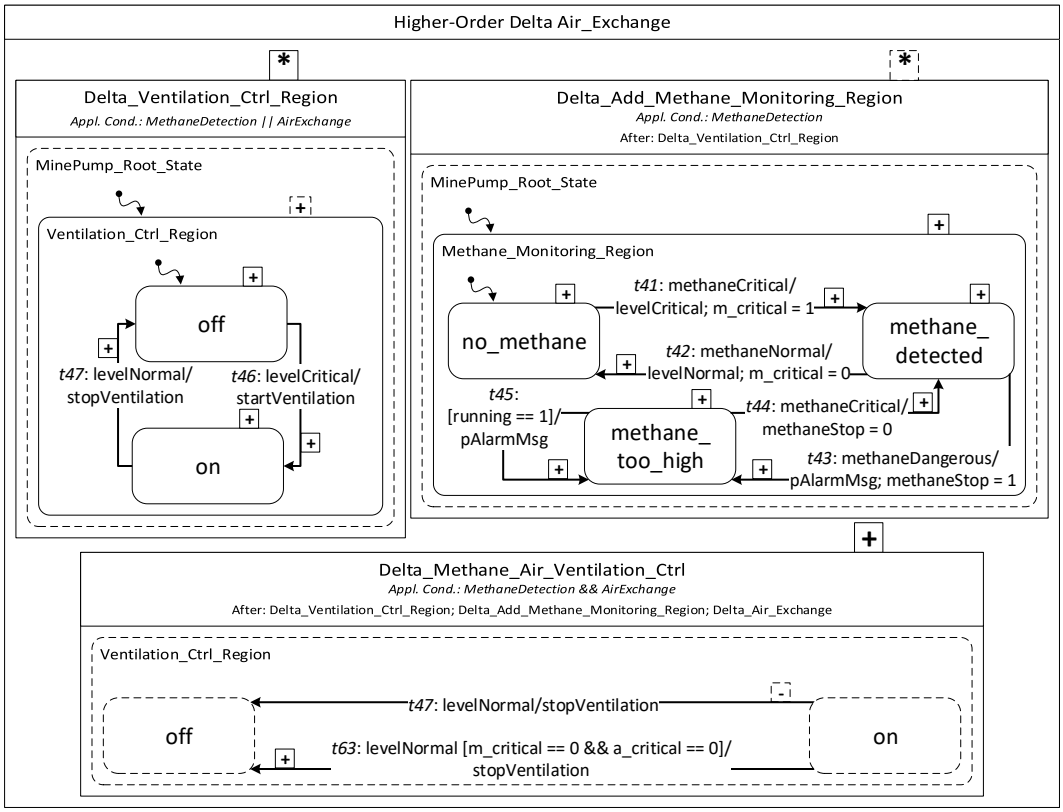

Abbildung 4.51.: Zweites Higher-Order Delta zum Hinzufügen eines Luftaustauschs

Durch dieses Szenario werden 16 Feature-Konfigurationen und somit auch 16 Produktvarianten ergänzt. Die Gesamtanzahl an Konfigurationen und Varianten beträgt damit je 48 Feature-Konfigurationen und Produktvarianten.

4.4. Body Comfort System

In diesem Abschnitt werden Evolutionsszenarien für das Body Comfort System beschrieben. Es folgen insgesamt vier Szenarien, welche das Ergänzen eines Scheibenwischers, die Erweiterung um elektrisch verstellbare Sitze mit Schlüssel-ID und das Hinzufügen von beheizbaren Scheiben und automatischem Licht umfassen.

4.4.1. Scheibenwischer

In diesem Szenario wird dem Body Comfort System ein Scheibenwischer hinzugefügt.

Beschreibung

Ausgangssituation: Das Body Comfort System besteht aus einer Türanlage mit elektrisch verstellbaren Spiegeln und elektrischen Fensterhebern, die manuell oder automatisch betätigt werden können, und einer Mensch-Maschine-Schnittstelle zur Interaktion, welche mit einer Menge von unterschiedlichen LEDs ausgestattet werden kann. Darüber hinaus besteht die Möglichkeit, verschiedene Sicherheitsfunktionen wie eine Zentralverriegelung, einen Funkschlüssel und eine Alarmanlage auszuwählen.

Szenario: Damit dem Kunden noch mehr Komfort geboten werden kann, sollen nun die standardmäßigen, manuellen Scheibenwischer durch eine automatische Scheibenwischanlage mit Sensor ersetzt werden. Um Konkurrenten auszuboten, wird daher die Firma des besten Scheibenwischanlagenherstellers auf dem Markt (aus Kapitel 4.2) aufgekauft und deren komplette Softwareproduktlinie in die Produktlinie des Body Comfort Systems integriert. Darüber hinaus sollen mehrere Status-LEDs zur Visualisierung einiger Scheibenwischerfunktionen ergänzt werden.

Prinzipiell könnten auch für das Body Comfort System die verschiedenen Funktionen der Scheibenwischanlage nach und nach in mehreren Szenarien eingebaut werden. Da diese Historie jedoch bereits in Kapitel 4.2 einmal komplett durchgespielt wurde, wird die gesamte Scheibenwischanlage hier als Komplettpaket in einem einzigen Szenario eingefügt.

Umsetzung: In Abbildung 4.52 ist das Feature-Diagramm des BCS zu sehen, welches um das gesamte Feature-Modell der Scheibenwischanlage aus Kapitel 4.2.5 erweitert wurde. Dazu wurde das Wurzel-Feature `Wiper` direkt als Mandatory-Kind-Feature von `Body Comfort System` angehängt. In jedem BCS-Produkt muss daher ab sofort immer eine Scheibenwischanlage mit mindestens der minderwertigen Sensor-Wischer-Kombination enthalten sein. Ebenfalls neu hinzugekommen sind die Or-Features `LED Wiper`, welche leuchtet, wenn die automatische Wischerfunktion aktiviert ist, `LED Frost Protection`, die anzeigt, wenn die Temperatur zu niedrig ist, um den Wischer zu verwenden, und `LED Clean` um zu signalisieren, dass der Scheibenreiniger aufgebraucht ist. `LED Clean` und `LED Frost Protection` benötigen die Features `Clean` beziehungsweise `Frost Protection` und sind daher mit diesen über eine `requires`-Relation verbunden.

Damit die Scheibenwischanlage im Body Comfort System umgesetzt werden kann, wird dem BCS die neue Substate Machine `Wiper` hinzugefügt, welche den gleichen Aufbau aufweist wie das Kernmodell der Scheibenwischanlage in Kapitel 4.2 und daher nicht gesondert im Produktmodell in Abbildung 4.53 dargestellt wird. Der Substate Machine `LED` werden drei neue Substate Machines zur Behandlung der LEDs für `Wiper Clean` und `Frost Protection` hinzugefügt. Die LED für `Frost Protection` springt an, sobald Frost erkannt wird. `LED_Clean` lässt die LED leuchten, wenn die Warnung über den niedrigen Scheibenreinigerstand gesendet wird. Ausgeschaltet wird diese LED

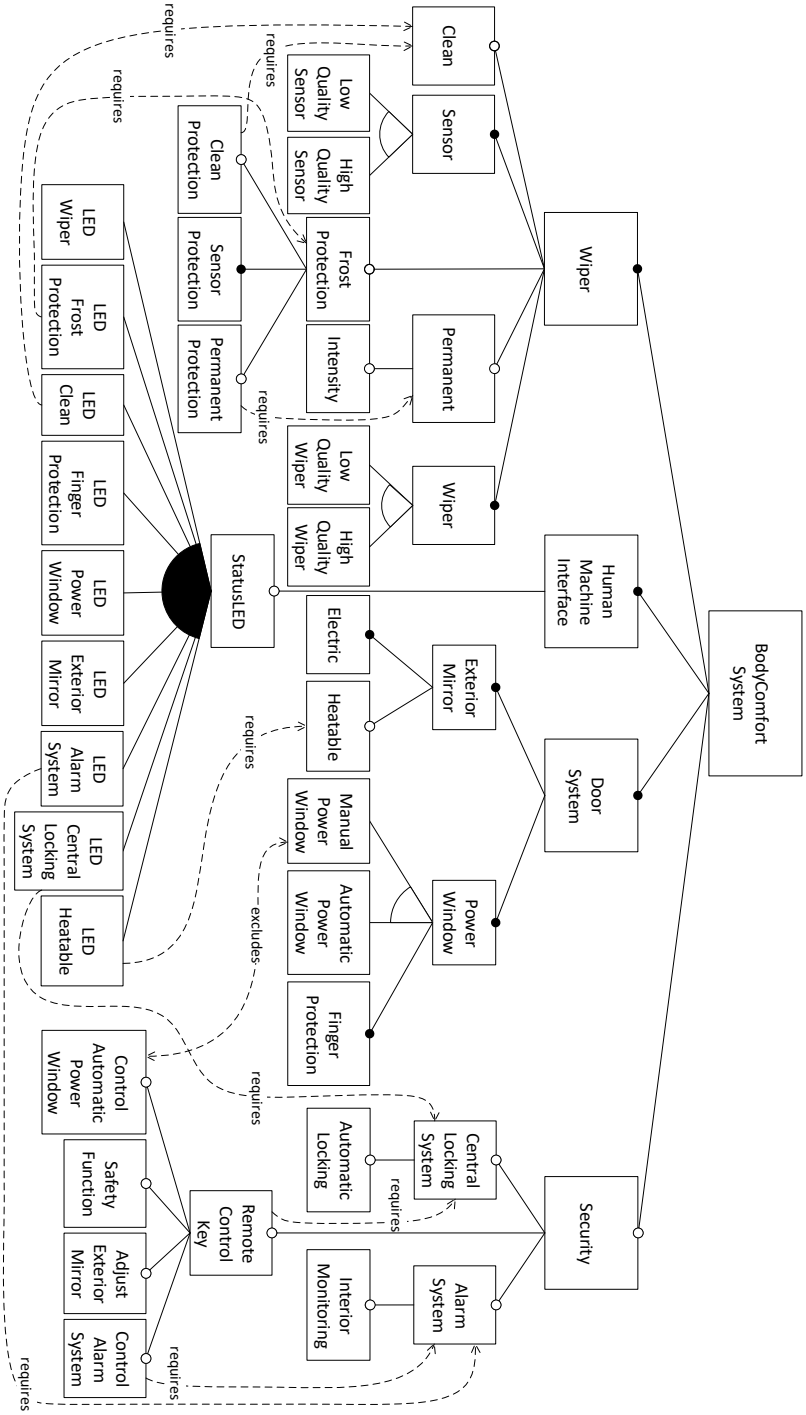

Abbildung 4.52.: Feature-Diagramm des BCS mit Scheibenwischer

erst dann wieder, wenn der Scheibenreiniger vollständig durch den Service aufgefüllt wurde. Die LED für `Wiper` ist genau dann eingeschaltet, wenn auch die automatische Wischfunktion aktiv ist, unabhängig davon, ob es gerade regnet oder nicht.

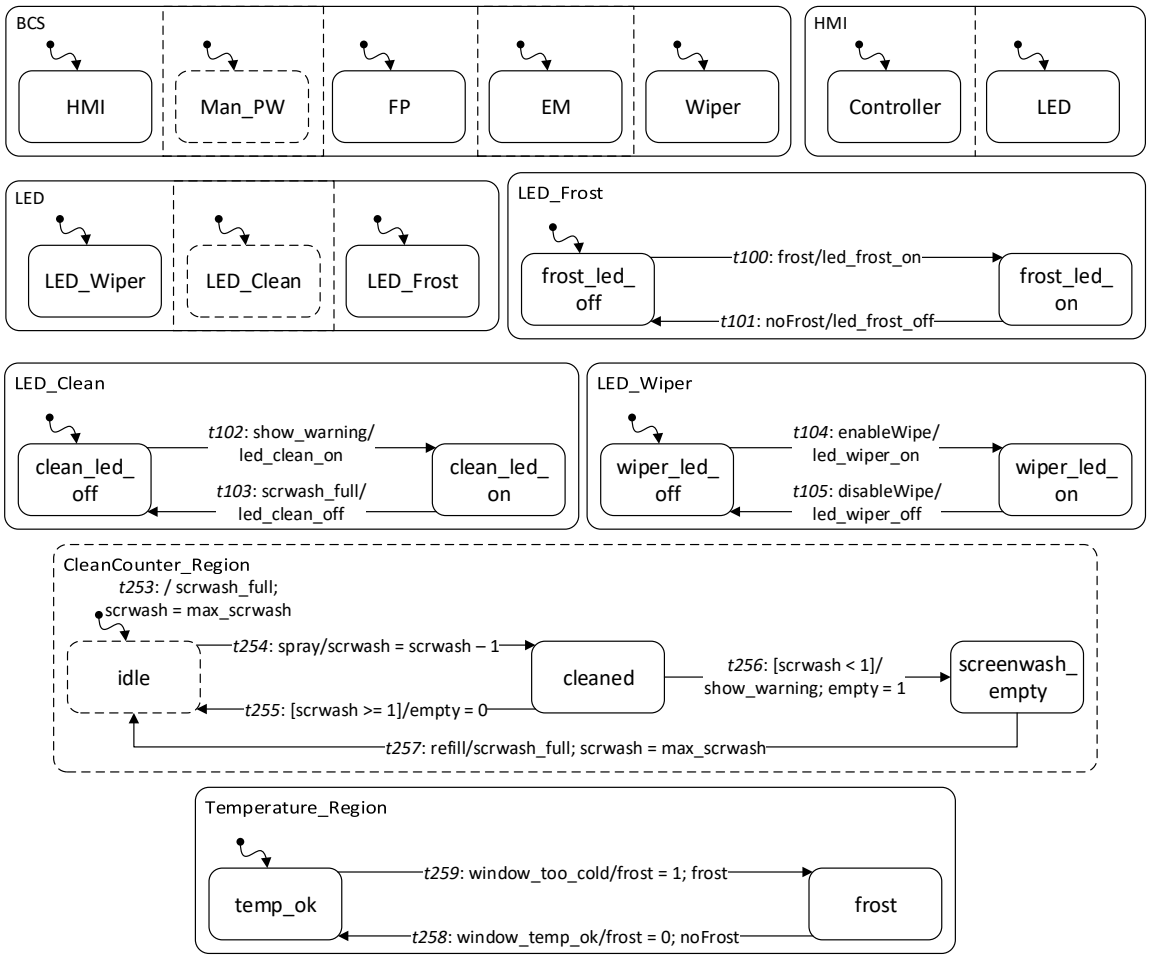

Abbildung 4.53.: Produktmodell für ein BCS mit Scheibenwischer

Modellierung

Die Deltas, die zur Umsetzung dieses neuen Features benötigt werden, sind ein neues Delta, welches das komplette Kernmodell aus Abbildung 3.5 einfügt, sowie sämtliche Deltas, die für das Szenario aus Kapitel 4.2.5 gültig waren, das heißt, nicht in vorhergehenden Szenarien entfernt wurden. Sämtliche Deltas der Scheibenwischanlage ohne Anwendungsreihenfolge müssen außerdem so angepasst werden, dass sie nur noch nach dem Kern-Delta (`Delta_Wiper_Core`) angewendet werden dürfen.

Darüber hinaus kommen drei weitere neue Deltas hinzu, welche im Higher-Order Delta `Wiper_LEDs` in Abbildung 4.54 dargestellt sind. `DAddStatusLEDFrost` ergänzt in `LED` die Substate Machine `LED_Frost` samt ihrer Zustände `frost_led_off` und `frost_led_on` sowie den Transitionen `t100` zum Einschalten der LED bei Frost und `t101` zum Abschalten der LED bei positiven Temperaturen. Dieses Delta darf nur nach `DAddStatusLED` und `Delta_Temperature` verwendet wer-

den. `DAddStatusLEDClean` fügt die Substate Machine `LED_Clean` hinzu, welche aus den Zuständen `clean_led_off` und `clean_led_on` und den Transitionen `102` zum Leuchten der LED bei zu wenig Scheibenreiniger und `t103` zum Ausschalten der LED bei aufgefülltem Scheibenreiniger besteht. Vor diesem Delta müssen zuerst `DAddStatusLED` und `Delta_Clean_Counter` benutzt werden. `DeltaAddStatusLEDWiper` fügt `LED_Wiper` ein. Diese Substate Machine setzt sich zusammen aus den Zuständen `wiper_led_off` und `wiper_led_on` und den Transitionen `t104`, welche bei Aktivieren der Automatik die LED einschaltet, und `t105`, welche die LED bei Deaktivieren der Automatik wieder ausschaltet. `Delta_LED_Wiper` kann erst nach `DAddStatusLED` und `Delta_Wiper_Core` angewendet werden.

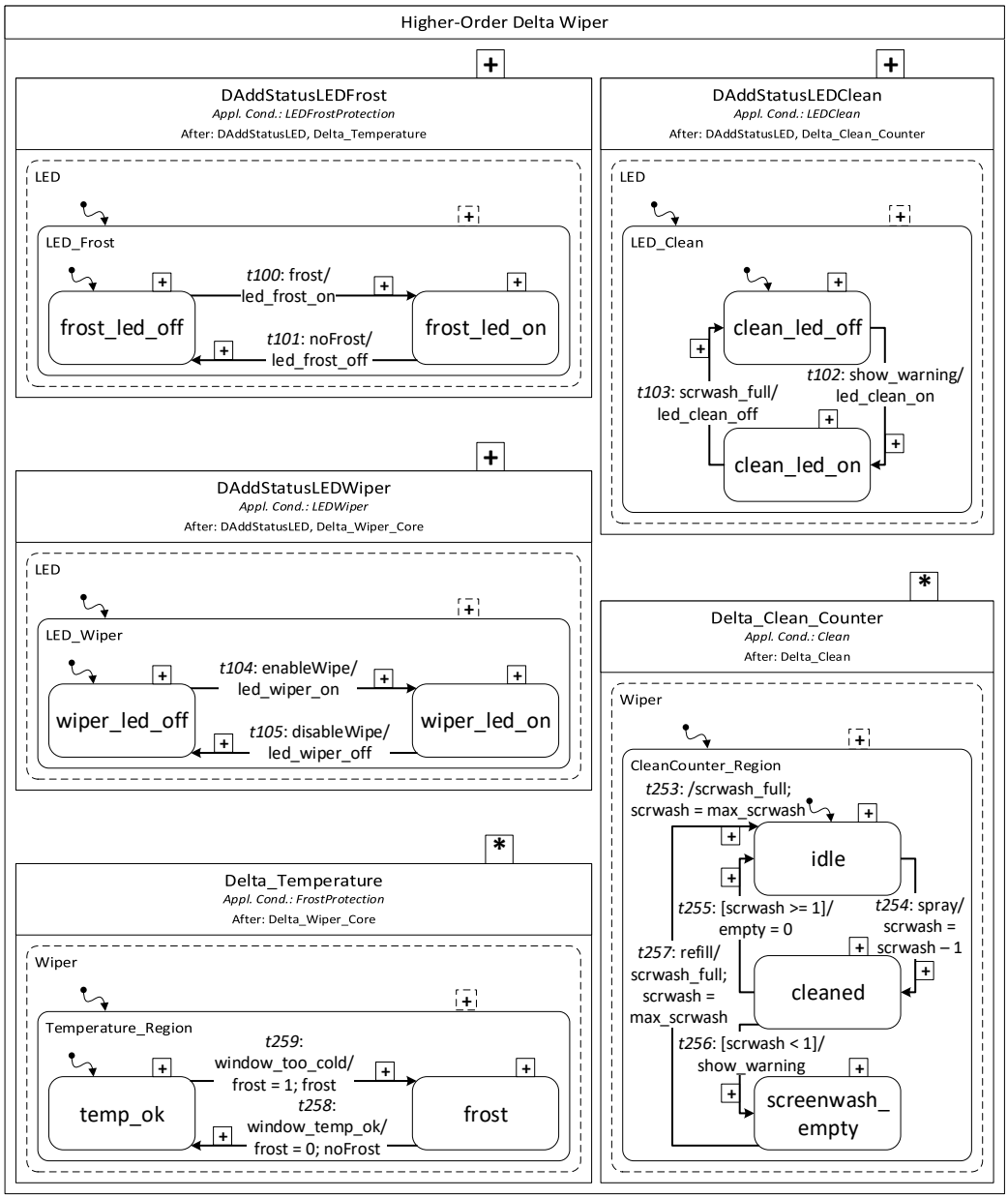

Abbildung 4.54.: Higher-Order Delta für das Hinzufügen eines Scheibenwischers

Des Weiteren werden in Wiper_LEDs die Deltas Delta_Temperature und Delta_Clean_Counter modifiziert, um an die neuen LEDs angepasst zu werden. Dazu werden von Delta_Temperature den Transitionen t257 und t258 die Aktionen frost beziehungsweise noFrost hinzugefügt. Die Transition t253 aus Delta_Clean_Counter wird um die Aktion scrwash_full ergänzt.

4.4.2. Elektrische Sitzeinstellung mit Schlüssel-ID

Dieses Szenario ergänzt das Body Comfort System um eine automatische, elektrische Sitzeinstellung anhand einer Schlüssel-ID.

Beschreibung

Ausgangssituation: Das Body Comfort System umfasst elektrisch verstellbare Spiegel elektrische Fensterheber und eine Scheibenwischanlage, kann unterschiedliche LEDs zur Anzeige von Informationen besitzen und mit Zentralverriegelung, Funkschlüssel und Alarmanlage ausgestattet werde.

Szenario: In vielen Haushalten, die sich ein Fahrzeug teilen, tritt das Problem auf, dass jedes Mal der Fahrersitz manuell verstellt werden muss, wenn zuvor jemand anderes gefahren ist. Vor allem, wenn sich zwei Personen häufig abwechseln, kann dies auf Dauer sehr lästig werden. Daher soll das Body Comfort System um eine Funktion erweitert werden, welche die ID des aktuell verwendeten Schlüssels erkennt und anhand derer die Sitzposition speichert. Pro Fahrzeug werden zwei Schlüssel geliefert.

Umsetzung: Das Feature-Diagramm des BCS, abgebildet in Abbildung 4.55, wird für dieses Szenario um das Optional-Feature Seat erweitert. Die elektrische Sitzverstellung funktioniert nur zusammen mit einem Funkschlüssel, daher ist Seat über eine requires-Relation mit dem Feature Remote Control Key verbunden.

Anhand der ID eines Schlüssels wird beim Abschließen die Sitzposition gespeichert, die unter Verwendung des Schlüssels eingestellt wurde. Beim nächsten Aufschließen mit dem Schlüssel wird die gespeicherte Sitzposition ausgelesen und der Fahrersitz entsprechend eingestellt, sollte er inzwischen verstellt worden sein. Um dieses Verhalten umzusetzen, wird das BCS um eine Substate Machine erweitert, die wiederum aus zwei weiteren Substate Machines besteht. Das resultierende Produktmodell mit den für das Szenario relevanten Teilen ist in Abbildung 4.56 dargestellt. Set_Seat lädt entsprechend der Schlüssel-ID die gespeicherte Position, verstellt wenn nötig den Sitz und speichert die Position später erneut. Seat_Pos verarbeitet die manuelle Verstellung des Sitzes in Vorwärts- oder Rückwärtsrichtung und speichert die eingestellte Sitzposition, welche der Schlüssel-ID beim Abschließen zugewiesen wird. Des Weiteren wird die Substate Machine der Zentralverriegelung angepasst, sodass beim Abschließen das Speichern der Sitzposition und beim Aufschließen je nach Schlüssel das Verstellen des Sitzes ausgelöst wird.

Modellierung

Durch das Higher-Order Delta Seat in den Abbildungen 4.57 und 4.58 werden elf neue Deltas hinzugefügt. DAddSeat fügt der State Machine BCS die leere Substate Machine Seat hinzu. Durch DAddSeatSet wird in Seat die Substate Machine Set_Seat hinzugefügt. Diese besteht aus den Zuständen keys_idle, key1_in_use und key2_in_use und den Transitionen t106 und t109, falls die gespeicherten Sitzpositionen von Schlüssel 1 beziehungsweise Schlüssel 2 der aktuellen Sitzposition entsprechen sollten, t107 und 110 zum Speichern der Sitzposition für Schlüssel 1 beziehungswei-

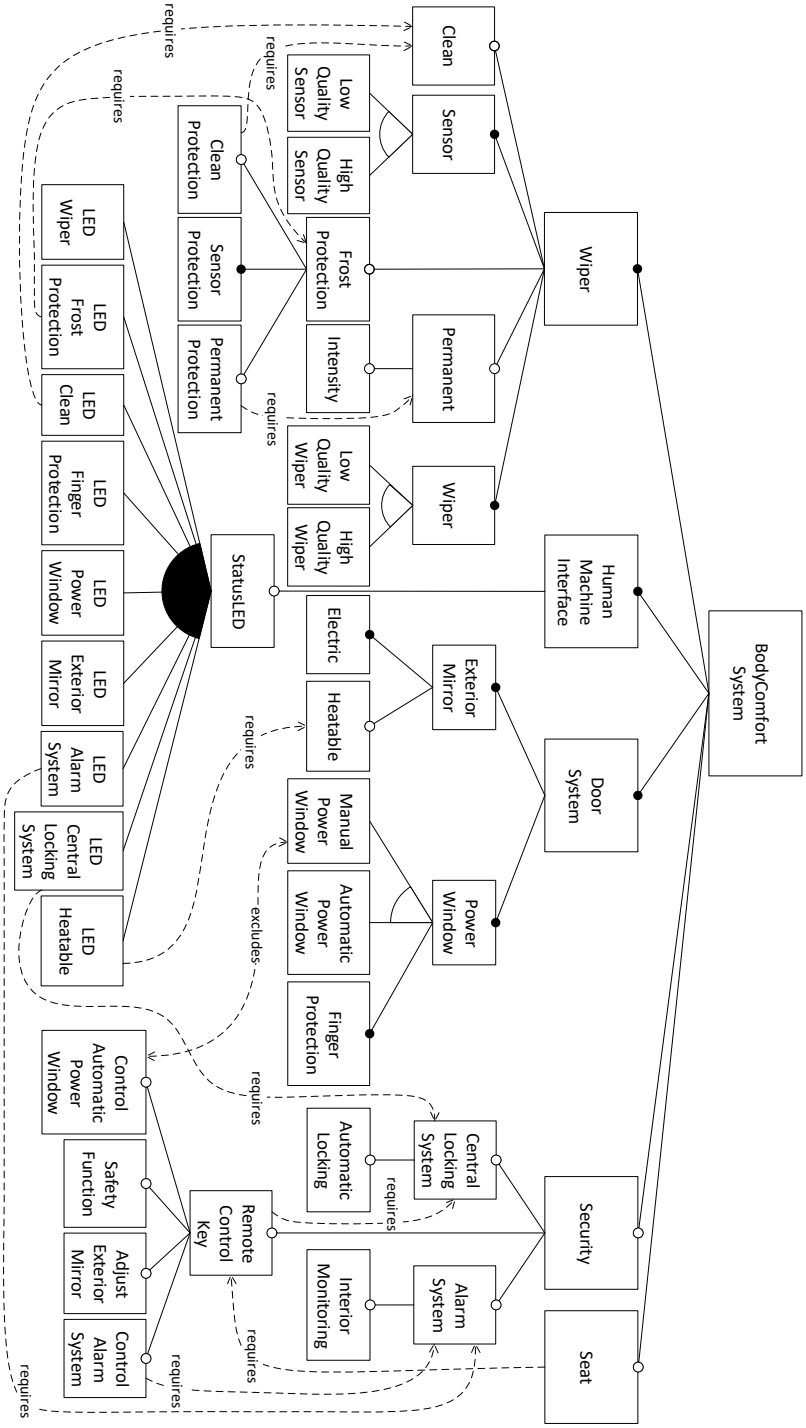

Abbildung 4.55.: Feature-Diagramm des BCS mit elektrischen Sitzen

se Schlüssel 2 und t108 und t111 zum Einstellen der jeweils gespeicherten Sitzposition, falls die gespeicherte Sitzposition nicht der aktuell eingestellten Position entsprechen sollte.

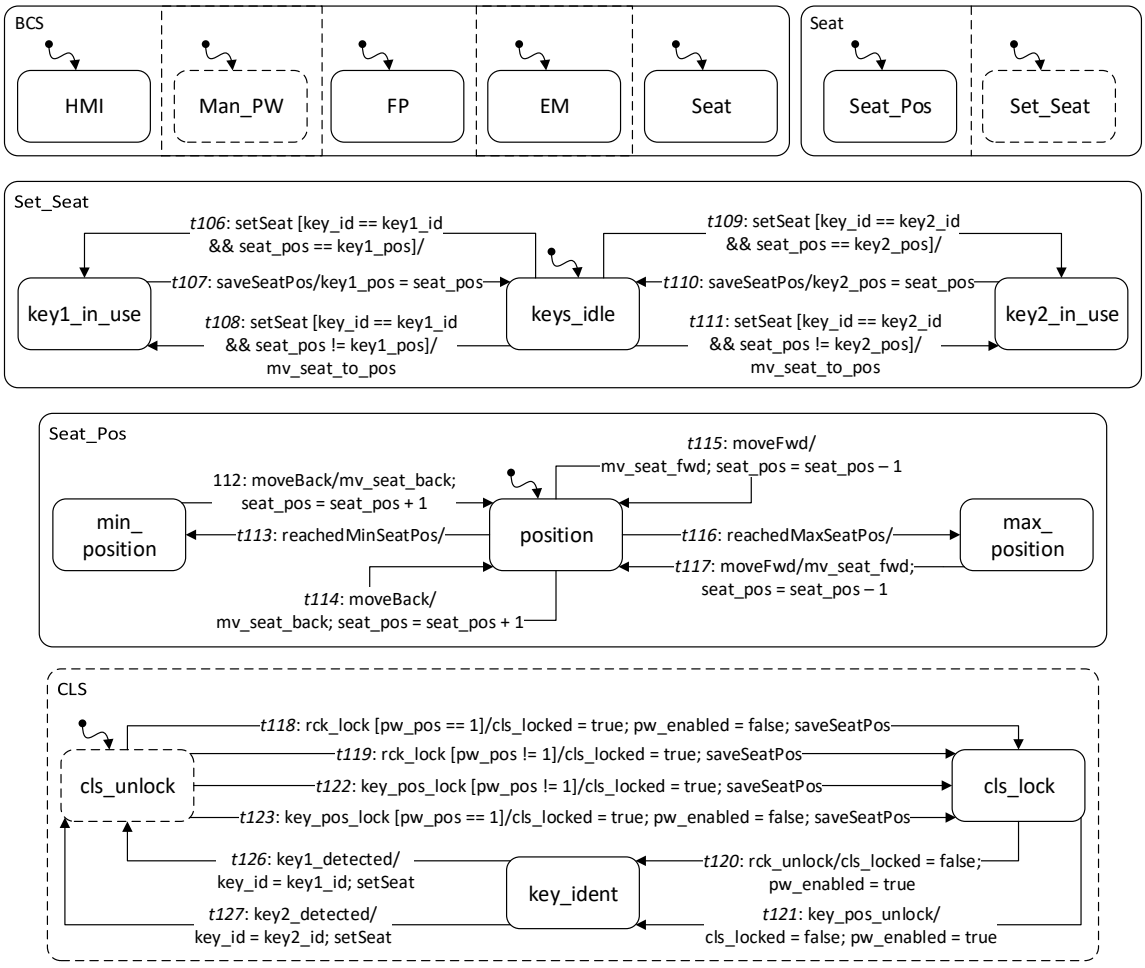

Abbildung 4.56.: Produktmodell für ein BCS mit elektrischen Sitzen

DAddSeatPos ergänzt in Seat die Substate Machine Seat_Pos. Diese setzt sich zusammen aus den Zuständen position, min_position und max_position sowie den Transitionen t112 zum Zurückschieben des Sitzes aus der vordersten Position, t113, wenn die vorderste Position erreicht ist, t114 zum Zurückschieben des Sitzes aus beliebiger, mittlerer Position, t115 zum Vorschieben des Sitzes aus beliebiger, mittlerer Position, t116, wenn die hinterste Position erreicht wird, und 117 zum Vorschieben des Sitzes aus der hintersten Position. Vor Anwendung von DAddSeatSet und DAddSeatPos muss zuerst DAddSeat verwendet werden.

Zusätzlich muss der Mechanismus für die Zentralverriegelung angepasst werden. DAddCLSBSM-ManPWSeat modifiziert dazu die Transitionen t49 und t50, DAddCLSBSMManPWRCKSeat die Transitionen t45 und t46, DAddCLSBSMAutoPWSeatKey die Transition t51 und DAddCLSBSMAutoPWRCK-SeatKey die Transition t52, um beim Abschließen die Speicherung der aktuellen Sitzposition auszulösen. Diese Deltas dürfen jeweils erst nach den Deltas angewendet werden, welche die ensprechenden, zu modifizierenden Transitionen einfügen.

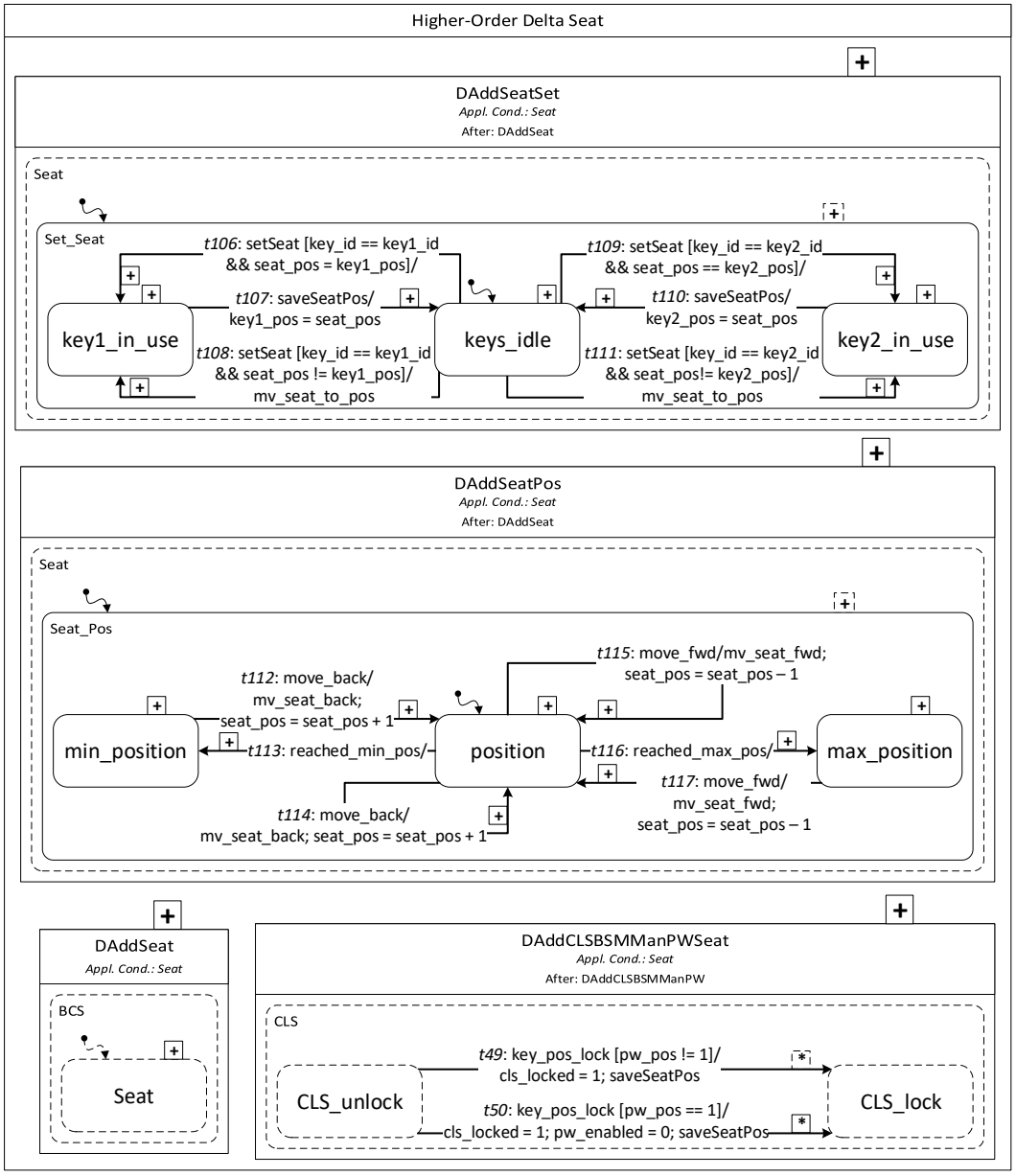

Abbildung 4.57.: Erstes Higher-Order Delta für das Hinzufügen von elektrischen Sitzen

Um beim Aufschließen die automatische Sitzeinstellung aufrufen zu können, muss durch `DAddSeatKeyIdent` der Zustand `key_ident` in `CLS` eingefügt werden. Vor `DAddSeatKeyIdent` muss zunächst `DAddCLS` angewendet werden. Durch `DAddCLSSBMSeatKey` wird dann Transition `t48` zwischen `CLS_lock` und `CLS_unlock` entfernt und zwischen `CLS_lock` und `key_ident` wieder eingefügt. Ebenso wird durch `DAddCLSSBMRCKSeat` mit der Transition `t47` verfahren. Delta `DAddCLSSBMSeat` darf erst nach den Deltas `DAddCLSSBM` und `DAddSeatKeyIdent` und Delta `DAddCLSSBMRCKSeat` erst nach `DAddCLSSBMRCK` und `DAddSeatKeyIdent` verwendet werden.

Zusätzlich werden durch `DAddCLSSBMSeatSetKey` die Transitionen `t118` und `t119` zwischen `key_ident` und `CLS_unlock` hinzugefügt. Diese beiden Transitionen sind dafür zuständig, zu erkennen, welcher Schlüssel verwendet wurde und diese Information für das Laden und Speichern der Sitz-

position weiterzugeben. Vor Verwendung von DAddCLSSBMSeatSetKey müssen zuerst die Deltas DAddSeatSet und DAddSeatKeyIdent benutzt werden.

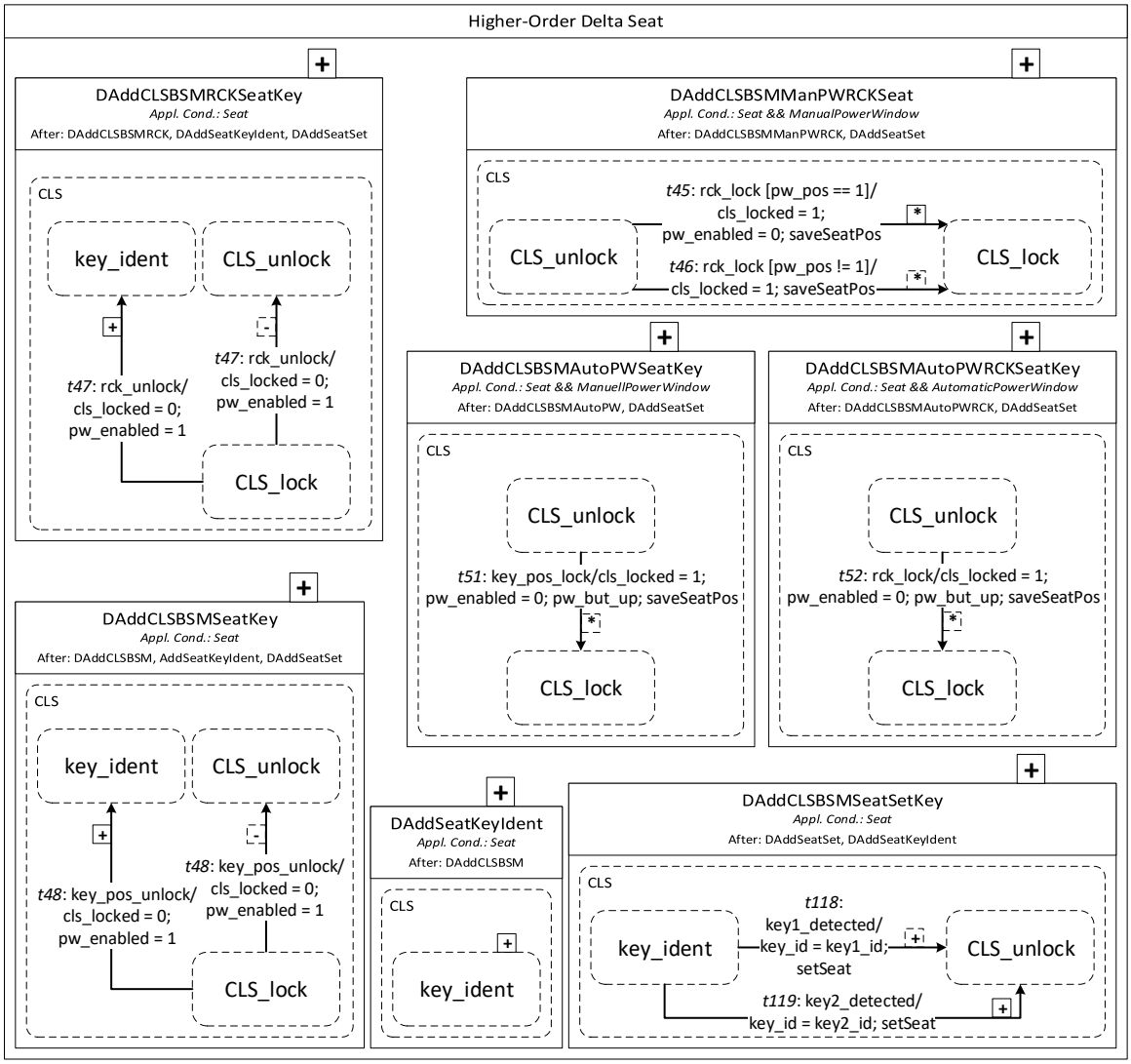

Abbildung 4.58.: Zweites Higher-Order Delta für das Hinzufügen von elektrischen Sitzen

4.4.3. Beheizbare Scheiben

In diesem Szenario wird das Body Comfort System um automatisch beheizbare Scheiben erweitert.

Beschreibung

Ausgangssituation: Das Body Comfort System setzt sich zusammen aus einem Türsystem mit elektrisch verstellbaren Spiegeln, die auf Wunsch beheizbar sind, und manuellen oder automatischen elektrischen Fensterhebern, aus einer Mensch-Maschine-Schnittstelle, die mit verschiedenen LEDs ausgestattet werden kann, und aus der Scheibenwischanlage aus Kapitel 4.2. Zusätzlich können ein automatisch verstellbarer Fahrersitz oder Sicherheitsfunktionen wie eine Zentralverriegelung, ein Funkschlüssel und eine Alarmanlage hinzugefügt werden.

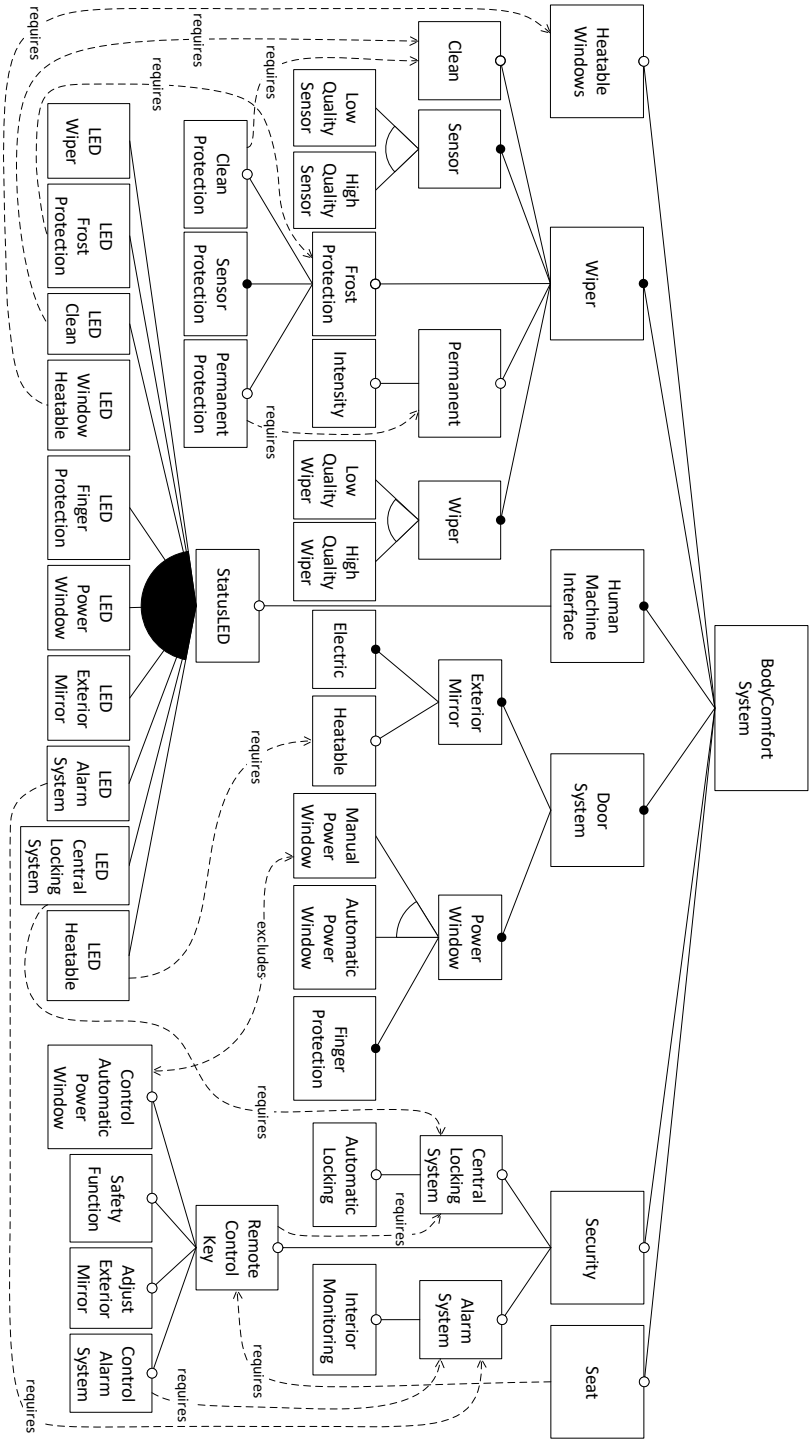

Abbildung 4.59.: Feature-Diagramm des BCS mit beheizbaren Scheiben

Szenario: Viele Kunden beschweren sich, dass es im Winter bei sehr kalten Temperaturen aufwändig ist, die Scheiben zu enteisen. Weiterhin dauere es eine ganze Weile bis die Frontscheibe eine positive Temperatur erreicht hat und für Produkte mit Frostüberprüfung dauere es daher unter Umständen sehr lang bis die Scheibenwischer einsatzfähig seien. Aus diesem Grund soll das Body Comfort System zukünftig um beheizbare Scheiben erweitert werden.

Umsetzung: Für diese Funktion wird dem Feature-Diagramm in Abbildung 4.59 das neue Optional-Feature `Heatable Windows` hinzugefügt. Zu diesem Feature kann auf Wunsch außerdem eine LED eingebaut werden, welche anzeigt, wenn die Scheibenheizung eingeschaltet ist. Aus diesem Grund wird zusätzlich das Feature `LED Window Heatable` ergänzt. `LED Window Heatable` benötigt immer das Feature `Heatable Windows`, weshalb diese beiden durch eine `requires`-Relation verbunden sind.

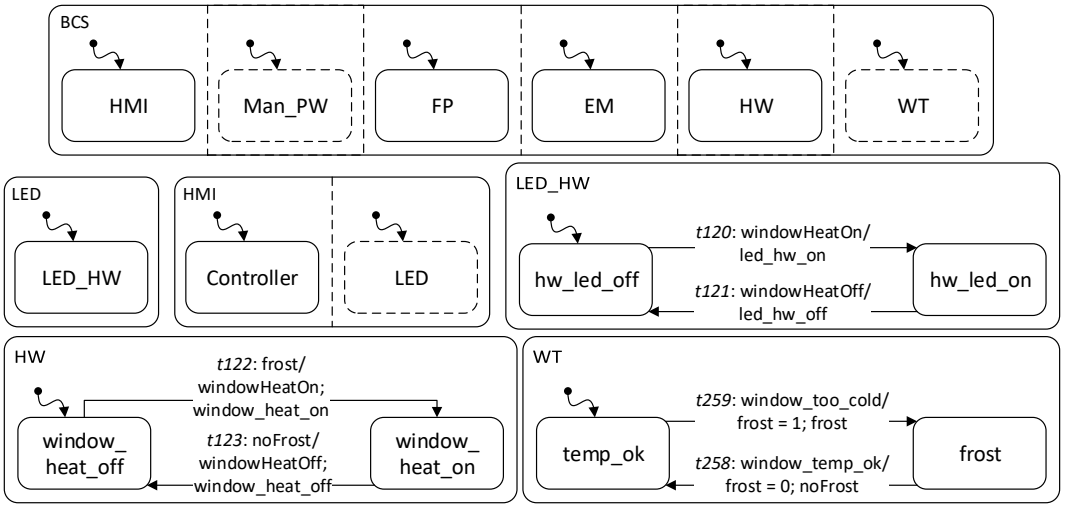

Abbildung 4.60.: Produktmodell für ein BCS mit beheizbaren Scheiben

Eine Funktion, welche die Scheibentemperatur überprüft und meldet, wenn diese zu kalt ist, existiert bereits im Zusammenhang mit der Scheibenwischanlage. Diese wird nun aus der Scheibenwischanlage ausgesondert und direkt als eigene Substate Machine `WT` (Window Temperature) in `BCS` eingegliedert, sodass sowohl das Feature `Heatable Windows` als auch das Feature `Frost Protection` darauf zugreifen können. Des Weiteren werden die Substate Machines `HW` (Heatable Windows) und `LED_HW` zum Ein- und Ausschalten der Scheibenheizung beziehungsweise der LED für die Scheibenheizung ergänzt.

Modellierung

Zur Umsetzung dieses Features müssen zwei neue Deltas zum Delta-Modell hinzugefügt und ein bestehendes Delta modifiziert werden. Dies geschieht durch das Higher-Order Delta `Heatable_Windows` in Abbildung 4.61. `DAddHW` fügt die Substate Machine `HW` in `BCS` hinzu. Diese besteht aus den beiden Zuständen `window_heat_off` und `window_heat_on` sowie den Transitionen `t122` und `t123`. Transition `t122` schaltet die Scheibenheizung ein, sobald die Scheibe als zu kalt erkannt wurde, und `t123` schaltet die Scheibenheizung aus, wenn die Temperatur der Scheibe wieder hoch genug ist. Dieses Delta darf erst nach `Delta_Temperature` angewendet werden.

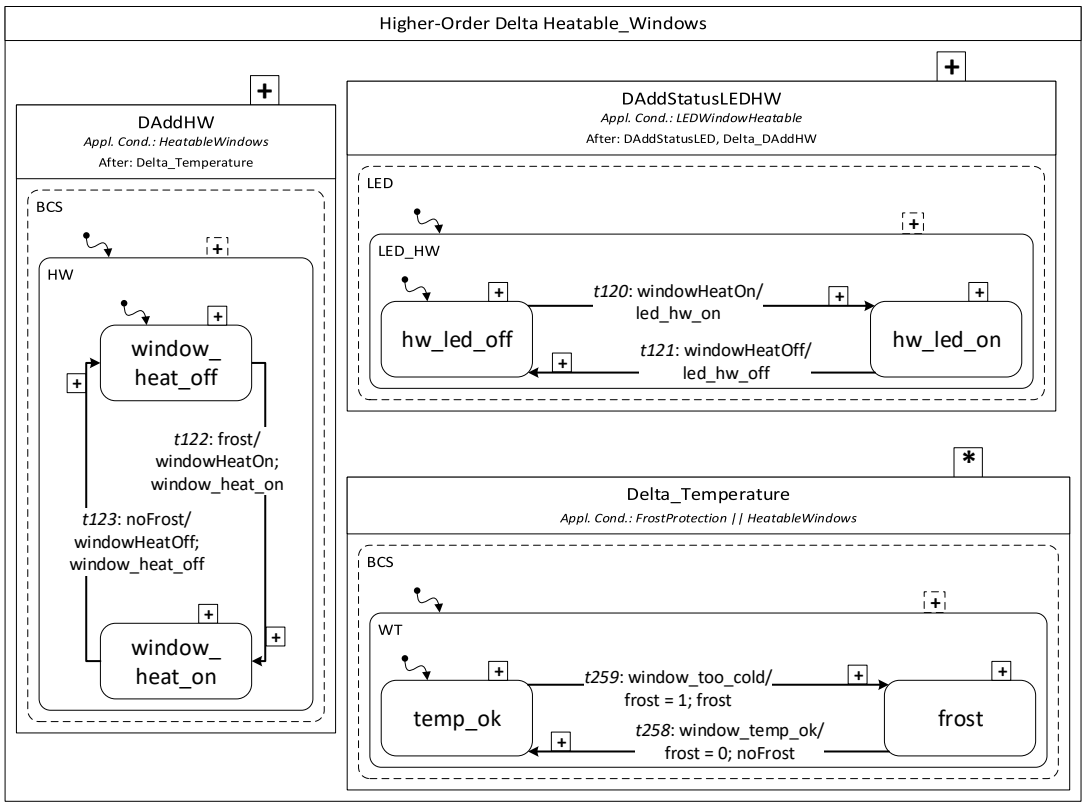

Abbildung 4.61.: Higher-Order Delta für das Hinzufügen von beheizbaren Scheiben

`DAddStatusLEDHW` ergänzt in `LED` die Substate Machine `LED_HW`. `LED_HW` setzt sich zusammen aus den Zuständen `hw_led_off` und `hw_led_on` und den Transitionen `t120` und `t121`. Transition `t120` schaltet die LED für die Scheibenheizung ein, wenn die Scheibenheizung läuft, und Transition `t121` schaltet die LED wieder aus, wenn die Scheibenheizung ausgestellt wird. Vor Anwendung von `DAddStatusLEDHW` müssen zunächst die Deltas `DAddStatusLED` und `DAddHW` benutzt werden.

`Delta_Temperature` wird modifiziert, sodass zum einen die Anwendungsbedingung erweitert wird und `Delta_Temperature` somit sowohl für `Frost Protection` als auch für `Heatable Windows` verwendet werden kann. Zum anderen werden die Zustände `temp_ok` und `frost` und die Transitionen `t259` und `t258` nun in `WT` eingefügt, welche als Substate Machine in `BCS` ergänzt wird. Damit gehört diese Substate Machine nicht mehr nur untergeordnet zum Feature `Wiper`, sondern besteht eigenständig.

4.4.4. Automatisches Licht

Dieses Szenario fügt dem Body Comfort System eine Funktion für automatisches Licht hinzu.

Beschreibung

Ausgangssituation: Das Body Comfort System enthält eine Scheibenwischanlage, eine Mensch-Maschine-Schnittstelle, elektrisch verstellbare Spiegel, die auf Wunsch beheizbar sein können, und manuell oder automatisch bedienbare, elektrische Fensterheber. Dazu kann es mit automatisch beheizbaren Scheiben, elektrisch verstellbaren Sitzen, Zentralverriegelung, Funkschlüssel, Alarmanlage und verschiedenen LEDs zur Anzeige von Informationen ausgestattet werden.

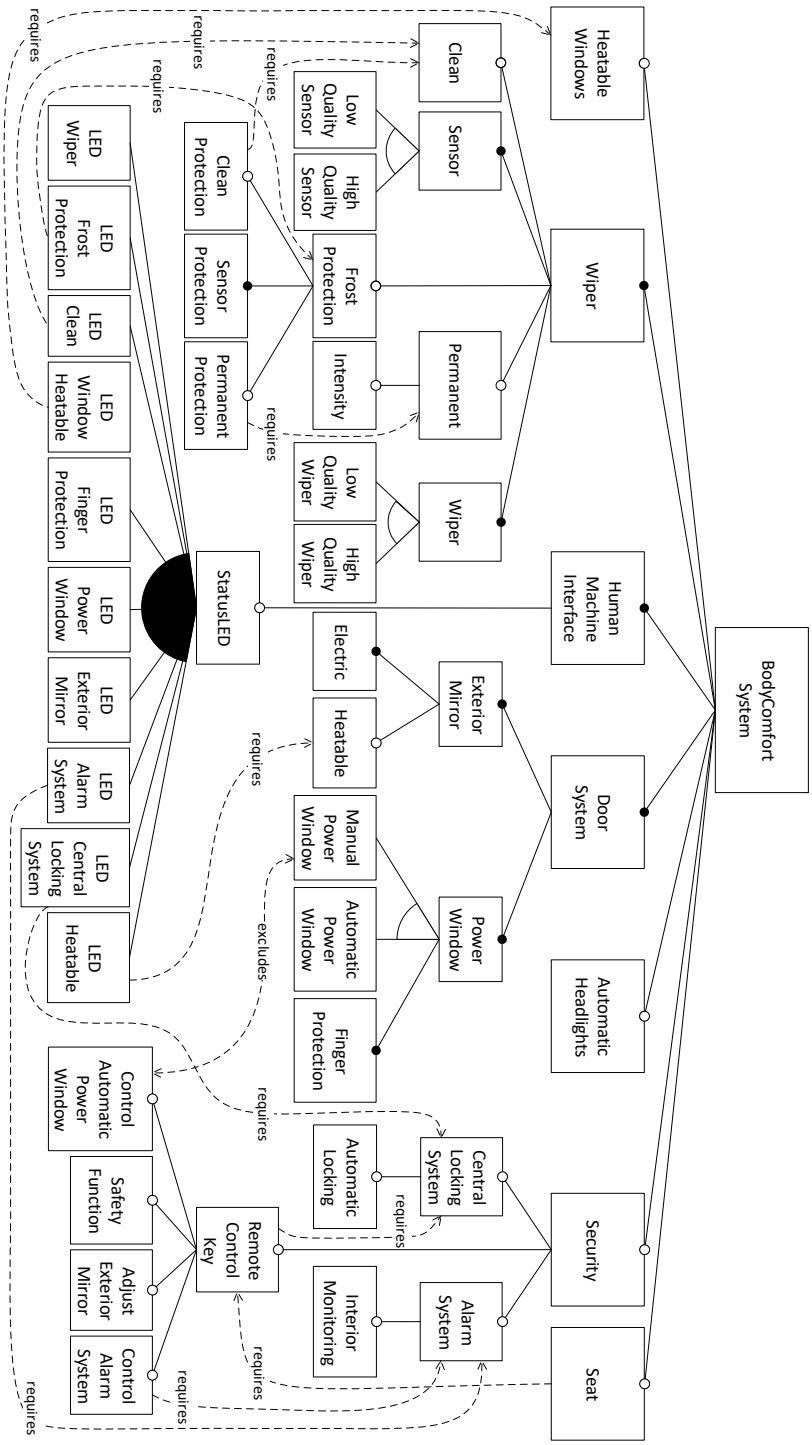

Abbildung 4.62.: Feature-Diagramm des BCS mit automatischem Licht

Szenario: Da den Fahrern häufig während der Fahrt erst zu spät auffällt, dass inzwischen Lichtverhältnisse erreicht sind, bei denen ein Einschalten des Lichtes angebracht ist, soll das Body Comfort System um eine automatische Lichtfunktion erweitert werden, welche das Licht automatisch den Tageslicht- und Fahrverhältnissen anpasst.

Umsetzung: Die automatische Lichtfunktion wird im Feature-Diagramm in Abbildung 4.62 umgesetzt, indem das Optional-Feature `Automatic Headlights` hinzugefügt wird.

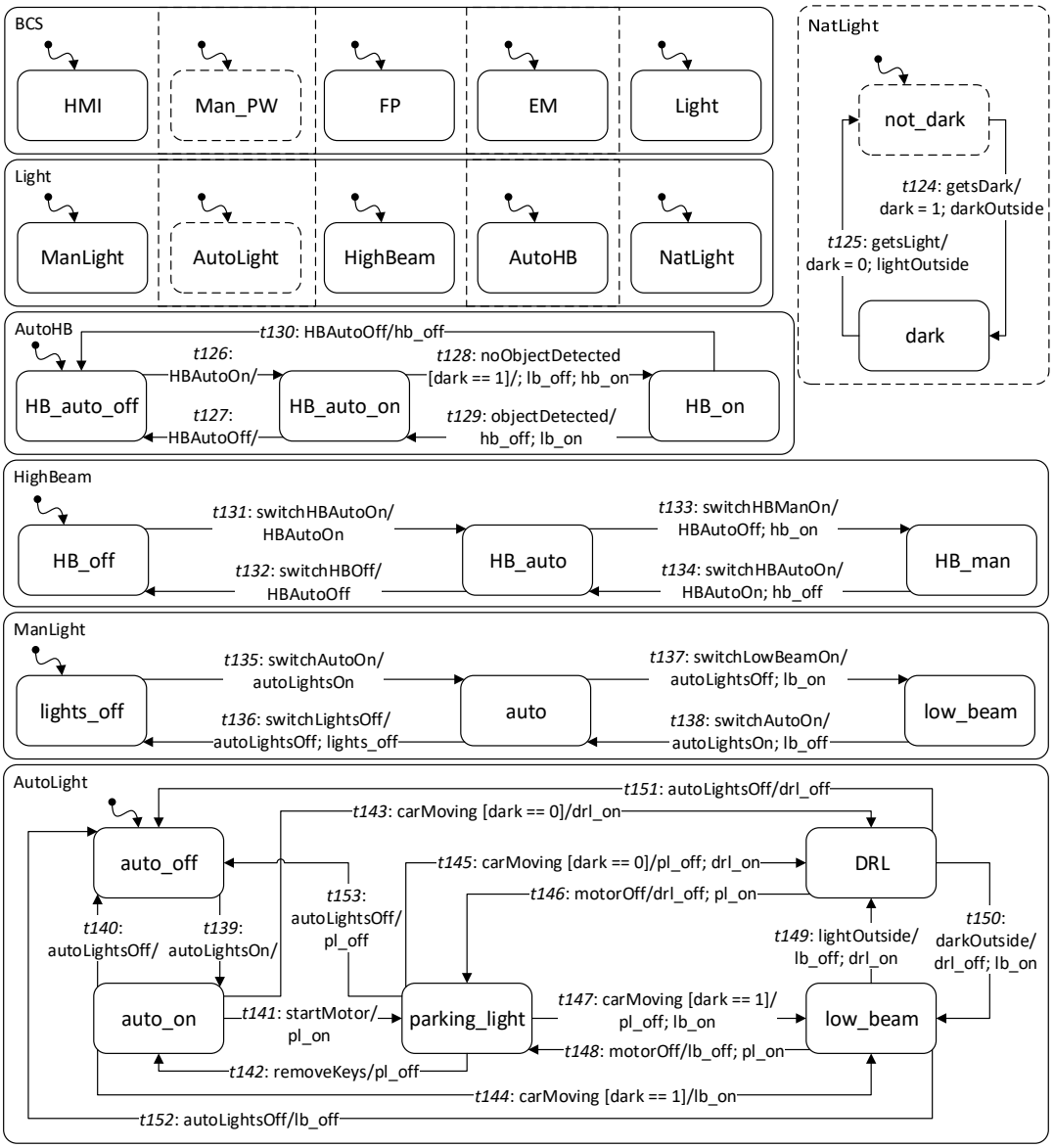

Abbildung 4.63.: Produktmodell für ein BCS mit automatischem Licht

Abbildung 4.63 zeigt ein Produktmodell mit der neuen Substate Machine `Light`, die wiederum aus fünf weiteren Substate Machines besteht. In `ManLight` kann manuell gewählt werden, ob das Licht ausgeschaltet, eingeschaltet oder auf automatisch gestellt sein soll. `NatLight` erhält eine Rückeldung über die Lichtverhältnisse draußen und entscheidet, ob diese als dunkel oder hell ein-

gestuft werden. `AutoLight` entscheidet je nach Situation, welches Licht angebracht ist und schaltet dieses ein. Bei Starten des Motors wird zunächst das Standlicht (engl. *parking lights*, PL) eingeschaltet, sobald das Fahrzeug losgefahren ist, wird bei ausreichend Licht das Tagfahrlicht (engl. *daytime running lights*, DRL) und bei Dunkelheit das Abblendlicht (engl. *low beam*, LB) eingeschaltet. Sollten die Lichtverhältnisse während der Fahrt wechseln, wird automatisch auf das entsprechende Licht umgeschaltet. Standlicht wird erst wieder verwendet, wenn der Motor ausgeschaltet wird, sodass bei kurzen Stopps, etwa an der Ampel, weiterhin Tagfahrlicht beziehungsweise Abblendlicht verwendet werden. Bei Entfernen der Schlüssel wird auch das Standlicht ausgeschaltet. Für den Fall, dass erst während der Fahrt das Licht manuell auf Automatik geschaltet wird, schaltet diese sofort entsprechend der Lichtverhältnisse Tagfahrlicht oder Abblendlicht ein. Darüber hinaus kann das Fernlicht (engl. *high beam*, HB) durch `HighBeam` manuell aus-, ein- oder auf Automatik geschaltet werden. Ist das Fernlicht auf Automatik gestellt, wird dieses automatisch eingeschaltet, sobald es dunkel ist und kein entgegenkommendes Fahrzeug erkannt wird. Das Fernlicht wird automatisch wieder ausgeschaltet, wenn ein Fahrzeug entgegen kommt.

Modellierung

Zur Umsetzung dieser Funktion werden elf neue Deltas benötigt, die durch das Higher-Order Delta `Automatic_Headlights` in den Abbildungen 4.64 und 4.65 hinzugefügt werden.

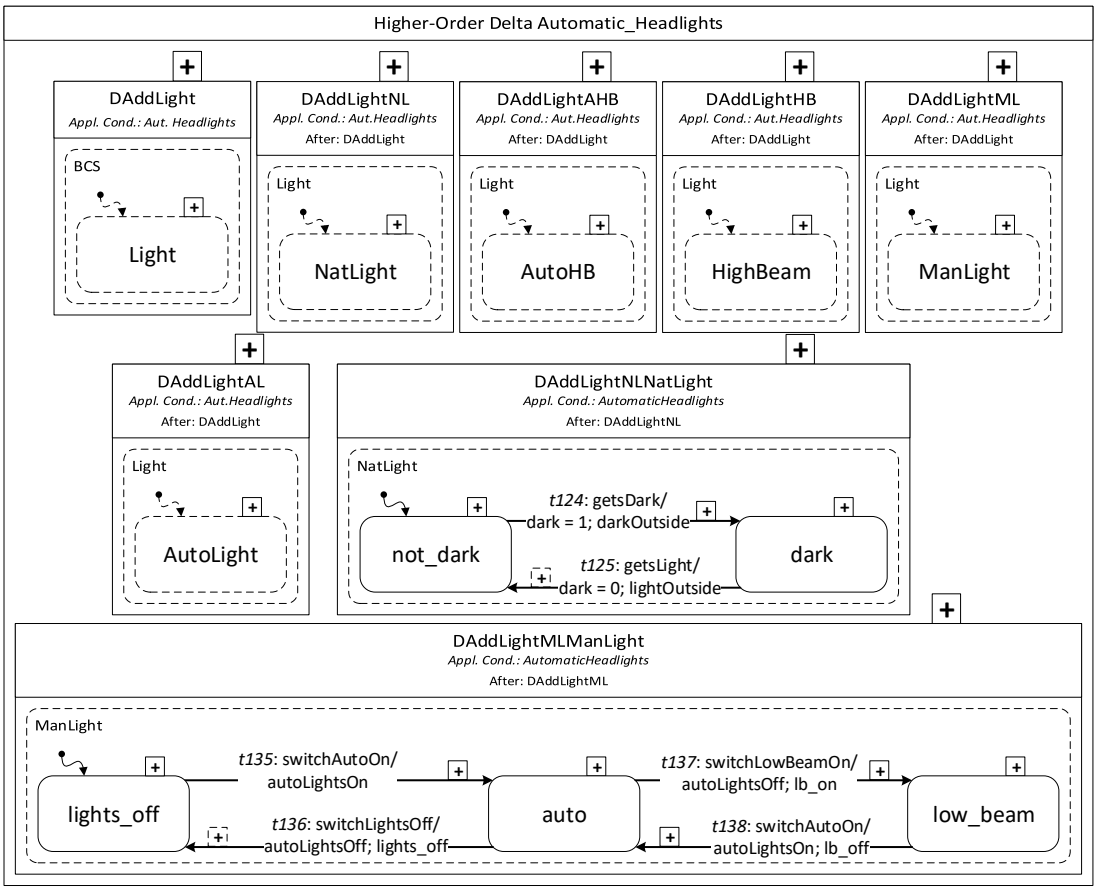

Abbildung 4.64.: Erster Teil des Higher-Order Deltas für das Hinzufügen von automatischem Licht

DAddLight ergänzt in BCS die leere Substate Machine Light. In dieser werden wiederum durch die Deltas DAddLightNL, DAddLightAHB, DAddLightHB, DAddLightML und DAddLightAL die Substate Machines NatLight, AutoHB, HighBeam, ManLight und AutoLight hinzugefügt. Diese fünf Deltas dürfen nur nach DAddLight verwendet werden.

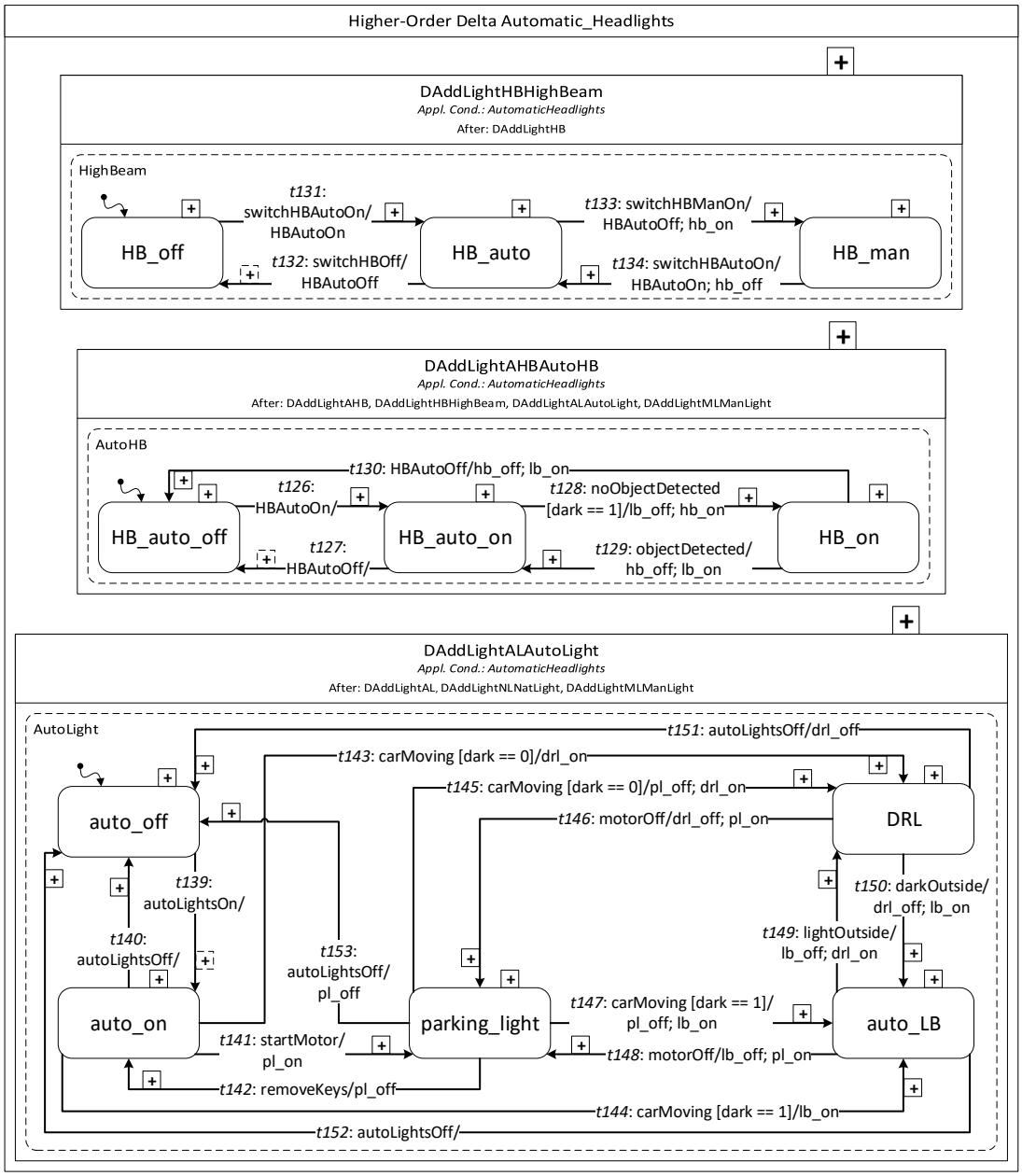

Abbildung 4.65.: Zweiter Teil des Higher-Order Deltas für das Hinzufügen von automatischem Licht

DAddLightNLNatLight fügt in NatLight die Zustände not_dark und dark sowie die Transitionen t124 zum Erkennen von Dunkelheit und t125 zum Erkennen von ausreichend Helligkeit ein. Vor diesem Delta muss zuerst DAddLightNL angewendet werden. DAddLightMLManLight ergänzt in ManLight die Zustände lights_off, auto und low_beam und die Transitionen t135

zum Einschalten der Automatik, t136 zum manuellen Ausschalten des Lichts, t137 zum manuellen Einschalten des Abblendlichts und t138 zum Wechseln von Abblendlicht auf Automatik. DAddLightMLManLight darf erst nach DAddLightML benutzt werden. DAddLightHBHighBeam fügt der Substate Machine HighBeam die Zustände HB_off, HB_auto und HB_man sowie die Transitionen t131 zum Einschalten der Fernlicht-Automatik, t132 zum manuellen Ausschalten des Fernlichts, t133 zum Einschalten von durchgängigem Fernlicht und 134 zum Schalten von Dauerfernlicht auf Fernlicht-Automatik hinzu. Vor Verwendung von DAddLightHBHighBeam muss zunächst DAddLightHB benutzt werden. DAddLightAHBAutoHB fügt in AutoHB die Zustände HB_auto_off, HB_auto_on und HB_on ein. Des Weiteren ergänzt DAddLightAHBAutoHB die Transitionen t126 zum Aktivieren der Fernlicht-Automatik, t127 und t130 zum Deaktivieren der Fernlicht-Automatik bei ausgeschaltetem beziehungsweise eingeschaltetem Fernlicht, t128 zum automatischen Einschalten des Fernlichts, falls Dunkelheit herrscht und kein entgegenkommendes Fahrzeug erkannt wurde, und 129 zum automatischen Ausschalten des Fernlicht, falls ein anderes Fahrzeug entgegenkommt. Dieses Delta darf erst nach den Deltas DAddLightAHB und DAddLightHBHighBeam verwendet werden.

DAddLightALAutoLight ergänzt in AutoLight die Zustände auto_off, auto_on, parking_light, DRL und low_beam. Außerdem fügt er insgesamt fünfzehn Transitionen hinzu. Transition t139 dient dabei zum Aktivieren der automatischen Lichtfunktion und t140 zum Deaktivieren. Transition t141 schaltet Standlicht an, wenn der Motor gestartet wird, t143 schaltet direkt nach Aktivieren Tagfahrlicht an, wenn das Fahrzeug sich bereits bewegt und es hell ist und t144 schaltet das Abblendlicht ein, wenn es dunkel ist und das Fahrzeug sich bereits bewegt. Transition t145 wechselt von Standlicht auf Tagfahrlicht, wenn es hell ist und das Fahrzeug losfährt, und t147 wechselt von Standlicht auf Abblendlicht, wenn es dunkel ist und das Fahrzeug losfährt. Die Transitionen t146 und t148 schalten von Tagfahrlicht beziehungsweise Abblendlicht wieder auf Standlicht, sobald der Motor ausgeschaltet wird und t142 schaltet das Standlicht aus, wenn der Schlüssel abgezogen wird. Mit den Transitionen t149 und t150 wird zwischen Tagfahr- und Abblendlicht gewechselt, wenn sich während der Fahrt die Lichverhältnisse ändern, und t151, t152 und t153 schalten Tagfahrlicht, Abblendlicht und Standlicht ab und die Automatik aus, wenn das Licht manuell ausgeschaltet wurde. Vor DAddLightALAutoLight müssen zuerst die Deltas DAddLightAL, DAddLightNLNatLight und DAddLightMLManLight verwendet werden.

5 Zusammenfassung und Fazit

Das Ziel dieser Arbeit waren die Entwicklung und Modellierung von Evolutionsszenarien für vier delta-orientierte Fallstudien im Kontext von Softwareproduktlinien. Die Evolutionsszenarien sollten dabei beschrieben und in einem prototypischen, Eclipse-basierten Modellierungswerkzeug umgesetzt werden.

Zu diesem Zweck wurden vorweg die benötigten Grundlagen für den Themenbereich dieser Arbeit vermittelt. Hierbei wurden zunächst Softwareproduktlinien [29] und deren Entwicklungsprozess vorgestellt. Des Weiteren wurden Feature-Modelle eingeführt, welche eine Möglichkeit bieten, sowohl die Gemeinsamkeiten als auch die Variabilität in einer Softwareproduktlinie grafisch darzustellen. Weiterhin wurde ein Überblick über verschiedene Arten der Variabilitätsmodellierung [32] von Artefakten gegeben. Mit Delta-Modellierung [7] wurde in diesem Zuge ein transformationaler Ansatz der Variabilitätsmodellierung genauer bekannt gemacht. Dieser wurde zudem im weiteren Verlauf dazu verwendet, die Fallstudien zu beschreiben, und diente als Grundlage für die in dieser Arbeit verwendete Methode zur Modellierung von Evolution. Abschließend für das Basiswissen wurden der Begriff der Evolution [23] und verschiedene Ansätze, Evolution zu modellieren, vorgestellt. Dabei wurde der Ansatz der Higher-Order Delta-Modellierung [21] genauer erläutert und dessen Funktionsweise vorgeführt.

Nachdem ein Überblick über die nötigen Vorkenntnisse gegeben wurde, wurden vier delta-orientierte Fallstudien eingeführt, welche Softwareproduktlinien beschreiben. Diese Fallstudien umfassten einen Verkaufsautomaten, eine Scheibenwischanlage und eine Minenpumpanlage [8] sowie ein Body Comfort System [22]. Zu jeder Fallstudie wurden sowohl das zugrunde liegende Feature-Modell als auch die Verhaltensspezifikation in Form einer State Machine ausführlich betrachtet. Die State Machines wurden als Delta-Modelle in Form von Kernmodell und initialen Deltas dargestellt.

Nach der Betrachtung der ursprünglichen Varianten der Fallstudien wurden im weiteren Verlauf verschiedene Evolutionsszenarien für die Fallstudien entwickelt. Dazu wurden die Auslöser und die Charakteristika der Veränderungen illustriert und die Auswirkungen der Veränderungen sowohl auf die Feature-Modelle als auch die Delta-Modelle beschrieben. Die Umgestaltung der Delta-Modelle wurde durch Higher-Order Delta-Modellierung in Form von Hinzufügen, Entfernen und Modifizieren von Deltas vorgenommen.

Für den Verkaufsautomaten ergaben sich sieben Evolutionsszenarien mit insgesamt 41 hinzugefügten, sechs entfernten und sechs modifizierten Deltas. Die Scheibenwischanlage erhält fünf Szenarien, welche 21 hinzugefügte und zwei modifizierte Deltas umfassen. Bei der Minenpumpanlage entstanden drei Evolutionsszenarien einschließlich sechzehn hinzugefügten, einem entfernten und fünf modifizierten Deltas. Das Body Comfort System lies sich in vier Szenarien um 52 hinzugefügte und ein modifiziertes Delta erweitern. Die Softwareproduktlinie des Verkaufsautomaten hat sich somit von 20 auf je 90 mögliche Feature-Konfigurationen und Produktvarianten erweitert, während die Anzahl der gültigen Konfigurationen und Varianten der Scheibenwischanlage

von acht auf 84 gestiegen ist, und die Minenpumpanlage umfasst nun je 48 statt anfänglicher 16 Feature-Konfigurationen und Produktvarianten. Das Body Comfort System besitzt mehr als 6000 Konfigurationen und Varianten.

Zusammenfassend kann somit gesagt werden, dass die ursprünglichen Softwareproduktlinien der Fallstudien je um eine Evolutionshistorie, gebildet aus mehreren Evolutionsszenarien, erweitert wurden und ihr Umfang an gültigen Konfigurationen dadurch erheblich gesteigert wurde. Alle Evolutionsszenarien wurden in dem Eclipse-basieren Prototyp, der von Lity et al. [21] vorgestellt wird, modelliert und ausführlich in dieser Arbeit dokumentiert.

Im Folgenden werden einige Erkenntnisse, die im Laufe der Arbeit hervorgetreten sind, erörtert. Des Weiteren wird ein Ausblick gegeben, in welcher Form sich die hier erarbeiteten Ergebnisse in folgenden Arbeiten nutzen lassen könnten.

Erkenntnisse

Durch die wiederholte Anwendung von Higher-Order Delta-Modellierung zur Umsetzung der Evolutionsszenarien konnten Rückschlüsse auf die Funktionalität und Anwendbarkeit dieses Ansatzes gezogen werden. Da die Beschreibung von Unterschieden zwischen zwei Objekten allgemein instinktiv durch das Aufzeigen von hinzugefügten, entfernten oder umgestalteten (modifizierten) Elementen geschieht, besteht mit Delta-Modellierung eine Möglichkeit zur Modellierung von Variabilität, die sich intuitiv anwenden lässt. Higher-Order Delta-Modellierung nutzt gleichsam Delta-Modellierung, um Unterschiede zwischen zwei verschiedenen Versionen von Delta-Modellen aufzuzeigen. Diese Unterschiede werden hierbei durch hinzugefügte, entfernte oder modifizierte Deltas herausgestellt. Durch diese gewohnheitsmäßige Kennzeichnung von Abweichungen zwischen zwei Modellversionen ist Higher-Order Delta-Modellierung leicht verständlich und auch für den ungeübten Benutzer spontan anwendbar. Darüber hinaus lassen sich auch komplexe Änderungen am Delta-Modell ohne Probleme durch die drei Operationen Hinzufügen, Entfernen und Modifizieren darstellen. Die vermeintlich kompliziertere Modifikation eines Deltas stellt ebenfalls keine Schwierigkeit dar. Die bereits bestehenden Inhalte eines Deltas in Form von Basisoperationen an Modellelementen werden dabei gleichermaßen lediglich entfernt beziehungsweise modifiziert oder es werden neue Inhalte hinzugefügt. Auch für Evolutionshistorien, die über einen Evolutionsschritt hinausgehen, gestatten Higher-Order Deltas eine intuitive Verwendung. Da ein Higher-Order Delta ausschließlich die Unterschiede zwischen zwei aufeinanderfolgenden Versionen eines Delta-Modells beschreibt, kann das Prinzip hierbei genauso angewendet werden wie für einen einzelnen Evolutionsschritt. Higher-Order Delta-Modellierung lässt sich demzufolge sehr gut nutzen, um Evolution in Softwareproduktlinien darzustellen.

Bei der Modellierung der in dieser Arbeit beschriebenen Evolutionsszenarien sind jedoch auch Probleme aufgetreten, welche allerdings nicht im Zusammenhang mit dem Ansatz der Higher-Order Delta-Modellierung stehen. Vielmehr bestanden die Schwierigkeiten mit der Modellierung durch die Funktionsweise des verwendeten Prototyps. Zunächst ist zu sagen, dass es sich um einen stabil funktionierenden, einfach zu bedienenden Prototypen handelt. Die wesentlichen Funktionen sind gut nutzbar umgesetzt und erlauben eine spontan verständliche Strukturierung der Higher-Order Delta-Modelle. Dennoch existieren kleinere Störfaktoren, welche die Effizienz bei der Erstellung und die Übersichtlichkeit des fertigen Ergebnisses leicht stören, die Anwendbarkeit im Großen und Ganzen allerdings nicht wesentlich beeinträchtigen:

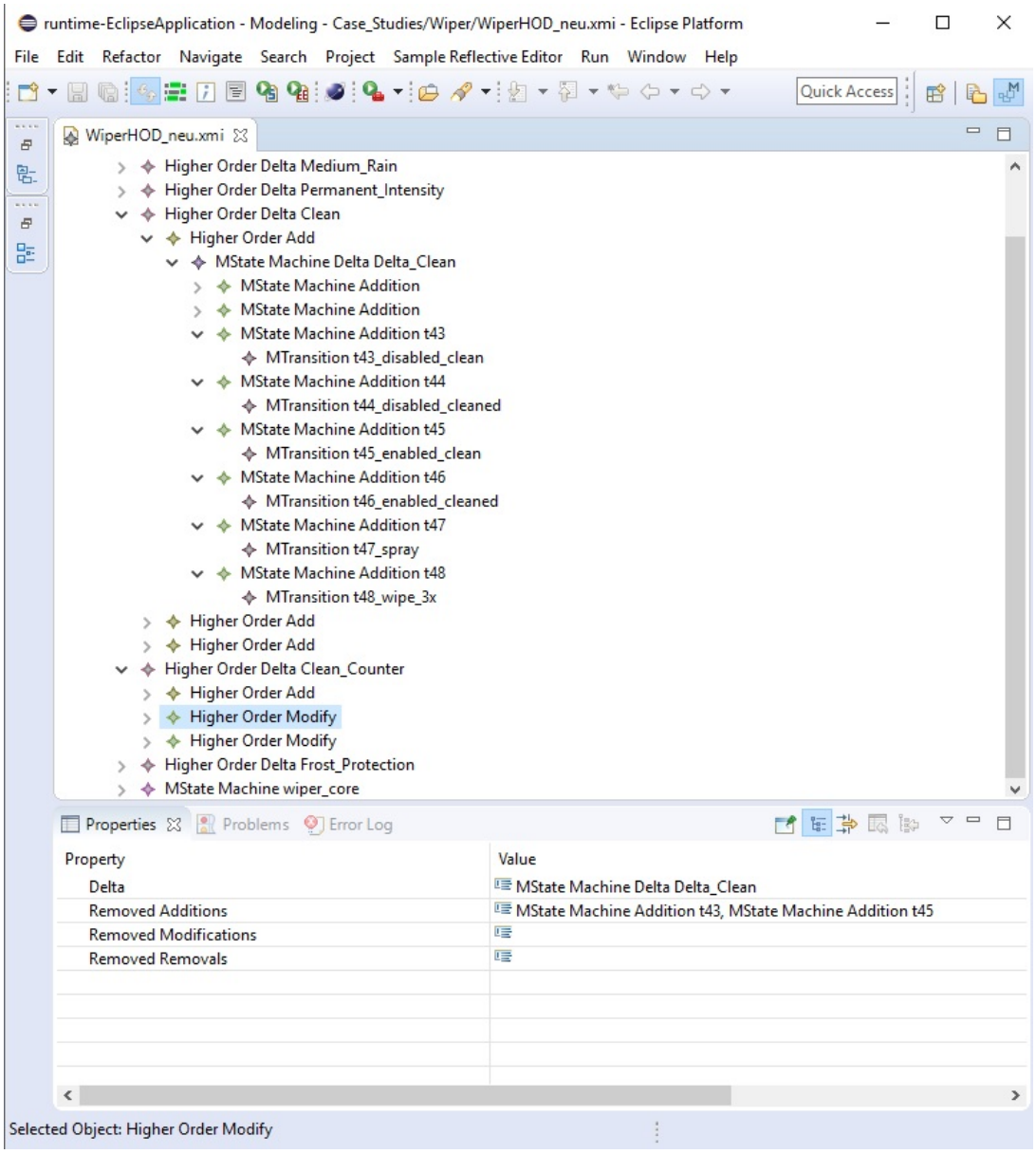

Abbildung 5.1.: Beispiel des Prototypen

1. Den Higher-Order-Additions, -Removals und -Modifys können keine Namen zugeordnet werden. Das ist im Wesentlichen nicht gravierend, stört jedoch, wenn in einem Higher-Order Delta sehr viele Higher-Order-Operationen existieren und eine bestimmte gesucht wird. In diesem Fall muss jedes Element einzeln selektiert und geöffnet werden, um ermitteln zu können, was sich darin verbirgt.

2. Dateiübergreifend können keine Inhalte kopiert werden. Diese Funktion wäre etwa nützlich gewesen, um die bereits für die Scheibenwischanlage erstellten Additions für das Body Comfort System wiederzuverwenden. Es war zwar möglich, die benötigten Elemente aus der XML-Quelldatei zu extrahieren und in der neuen Datei an der richtigen Stelle einzufügen, für sehr große Dateien wäre dieses Vorgehen allerdings zu ineffizient und fehleranfällig. Dieser Um-

stand ist jedoch eher dem Eclipse Modeling Framework geschuldet, auf dessen Grundlage der Prototyp basiert, als der Umsetzung des Prototypen selbst.

3. Higher-Order-Modifys bieten keine Möglichkeit Transitionen oder Zustände einzeln zu entfernen. Wird ein neues Delta durch eine Higher-Order-Addition erstellt, werden die im Delta enthaltenen Operationen in Delta-Additions, -Removals und -Modifys untergeordnet. Eine Delta-Addition kann dabei aus mehreren Transitionen, Zuständen, Events und ähnlichem bestehen. In einem Higher-Order-Modify kann nur eine komplette Delta-Addition entfernt werden, das bedeutet alle enthaltenen Zustände, Transitionen und weiteren Elemente, werden entfernt. Dieser Umstand ist in Abbildung 5.1 dargestellt. Dies erfordert also entweder ein hohes Maß an Vorausplanung oder einen störenden Mehraufwand. Durch die Vorausplanung könnten alle Elemente, die in einem späteren Schritt durch ein Higher-Order-Modify entfernt werden sollen, in eine gemeinsame Delta-Addition eingeordnet werden. Da bei Softwareproduktlinien allerdings nicht immer alle Änderungen vorausgesehen werden können, wäre dieses Vorgehen nicht alltagstauglich. Die andere Möglichkeit wäre, jedes einzelne Element, das hinzugefügt werden soll, seien es Transitionen, Zustände, Events oder andere, in eine eigene Delta-Addition einzufügen (siehe Abbildung 5.1). Dies erhöht allerdings die Anzahl an Schritten zur Erstellung eines neuen Deltas immens und beeinträchtigt somit stark die Effizienz.

4. Es ist keine grafische Darstellung der Modellierung möglich. Dies würde noch einmal eine schönere Gesamtübersicht und ein schnelleres Verständnis der Inhalte erlauben.

Alles in allem lässt sich somit sagen, dass Higher-Order Delta-Modellierung eine angenehme und einfach zu verwendende Methode zur Modellierung von Evolution in Softwareproduktlinien ist. Der Eclipse-basierte Prototyp bildet eine gute Grundlage, um mit Higher-Order Delta-Modellierung werkzeuggestützt Evolution modellieren zu können, besitzt jedoch einige Kritikpunkte, die eine effiziente Verwendung und übersichtliche Darstellung bei großen Softwareproduktlinien beeinträchtigen.

Weiterhin konnte beobachtet werden, dass Modifikationen von Deltas nicht zwangsläufig zu Modifikationen von Produktvarianten führen. Vielmehr fand eine Umgestaltung bereits existierender Varianten immer dann statt, wenn die Funktion von schon zuvor vorhandenen Features verändert wurde. So können auch entfernte oder neu hinzugefügte Deltas in einer Modifikation von Produktmodellen resultieren, sofern diese Deltas als Anwendungsbedingung ein Feature enthalten, welches sowohl vor als auch nach dem Evolutionsschritt zur Auswahl steht.

Ausblick

Die neu geschaffenen Evolutionshistorien der Fallstudien können verwendet werden, um neue Methoden zur Modellierung von Evolution zu evaluieren. Dabei können sie nicht nur für deltaorientierte Vorgehensweisen genutzt werden, sondern bei Bedarf auch in passende Formate für andere transformationale oder annotative und kompositionale Ansätze umgewandelt werden. In diesem Kontext können die Szenarien sowohl eingesetzt werden, um einzelne Methoden für sich selbst auf Funktionalität, Korrektheit und Anwendbarkeit zu prüfen, als auch Vergleiche zwischen unterschiedlichen Ansätzen zu ziehen. So könnte etwa kontrolliert werden, ob verschiedene Vorgehensweisen für gleiche Feature-Konfigurationen übereinstimmende Produktmodelle liefern. Ebenso

könnten Ansätze hinsichtlich ihres Zeitverhaltens und ihrer Effizienz bei der Erstellung dieser Produktmodelle verglichen werden.

Schlussendlich kann folglich festgehalten werden, dass sich die Evolustionsszenarien dieser Arbeit in vielfältiger Weise bei der Evaluation von Ansätzen zur Evolutionsmodellierung nutzen lassen.

Literatur

[1] M. Acher, P. Heymans, P. Collet, C. Quinton, P. Lahire und P. Merle. "Feature Model Differences". In: *24th International Conference on Advanced Information Systems Engineering(CAiSE'12)*. LNCS. Springer, 06/2012, S. 629–645. URL: https://nyx.unice.fr/publis/acher-heymans-etal:2012.pdf.

[2] V. Alves, R. Gheyi, T. Massoni, U. Kulesza, P. Borba und C. Lucena. "Refactoring Product Lines". In: *Proceedings of the 5th International Conference on Generative Programming and Component Engineering*. GPCE '06. Portland, Oregon, USA: ACM, 2006, S. 201–210. ISBN: 1-59593-237-2. DOI: 10.1145/1173706.1173737. URL: http://doi.acm.org/10.1145/1173706.1173737.

[3] J. Becker, W. Probandt und O. Vering. "Modellierung". In: *Grundsätze ordnungsmäßiger Modellierung: Konzeption und Praxisbeispiel für ein effizientes Prozessmanagement*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, S. 1–3. ISBN: 978-3-642-30412-5. DOI: 10.1007/978-3-642-30412-5_1. URL: http://dx.doi.org/10.1007/978-3-642-30412-5_1.

[4] D. Benavides, S. Segura und A. Ruiz-Cortés. "Automated Analysis of Feature Models 20 Years Later: A Literature Review". In: *Inf. Syst.* 35.6 (09/2010), S. 615–636. ISSN: 0306-4379. DOI: 10.1016/j.is.2010.01.001. URL: http://dx.doi.org/10.1016/j.is.2010.01.001.

[5] G. Botterweck und A. Pleuss. "Evolution of Software Product Lines". In: *Evolving Software Systems*. Hrsg. von T. Mens, A. Serebrenik und A. Cleve. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, S. 265–295. ISBN: 978-3-642-45398-4. DOI: 10.1007/978-3-642-45398-4_9. URL: http://dx.doi.org/10.1007/978-3-642-45398-4_9.

[6] G. Botterweck, A. Pleuss, D. Dhungana, A. Polzer und S. Kowalewski. "EvoFM: Feature-driven Planning of Product-line Evolution". In: *Proceedings of the 2010 ICSE Workshop on Product Line Approaches in Software Engineering*. PLEASE '10. Cape Town, South Africa: ACM, 2010, S. 24–31. ISBN: 978-1-60558-968-8. DOI: 10.1145/1808937.1808941. URL: http://doi.acm.org/10.1145/1808937.1808941.

[7] D. Clarke, M. Helvensteijn und I. Schaefer. "Abstract Delta Modeling". In: *SIGPLAN Not.* 46.2 (10/2010), S. 13–22. ISSN: 0362-1340. DOI: 10.1145/1942788.1868298. URL: http://doi.acm.org/10.1145/1942788.1868298.

[8] A. Classen. *Modelling with FTS: A Collection of Illustrative Examples*. Techn. Ber. P-CS-TR SPLMC-00000001. PReCISE Research Center, Univ. of Namur, 2010.

[9] P. Clements und L. Northrop. *Software Product Lines: Practices and Patterns*. Addison-Wesley Professional, 2001.

[10] D. Dhungana, P. Grünbacher, R. Rabiser und T. Neumayer. "Structuring the Modeling Space and Supporting Evolution in Software Product Line Engineering". In: *J. Syst. Softw.* 83.7 (07/2010), S. 1108–1122. ISSN: 0164-1212. DOI: 10.1016/j.jss.2010.02.018. URL: http://dx.doi.org/10.1016/j.jss.2010.02.018.

[11] A. Fantechi und S. Gnesi. "Formal Modeling for Product Families Engineering". In: *Software Product Line Conference, 2008. SPLC '08. 12th International*. 09/2008, S. 193–202. DOI: 10.1109/SPLC.2008.45.

[12] A. Gruler, M. Leucker und K. Scheidemann. "Modeling and Model Checking Software Product Lines". In: *Formal Methods for Open Object-Based Distributed Systems: 10th IFIP WG 6.1 International Conference, FMOODS 2008, Oslo, Norway, June 4-6, 2008 Proceedings*. Hrsg. von G. Barthe und F. S. de Boer. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, S. 113–131. ISBN: 978-3-540-68863-1. DOI: 10.1007/978-3-540-68863-1_8. URL: http://dx.doi.org/10.1007/978-3-540-68863-1_8.

[13] A. Haber, H. Rendel, B. Rumpe und I. Schaefer. "Delta Modeling for Software Architectures". In: *CoRR* abs/1409.2358 (2014). URL: http://arxiv.org/abs/1409.2358.

[14] A. Haber, H. Rendel, B. Rumpe und I. Schaefer. "Evolving Delta-Oriented Software Product Line Architectures". In: *Large-Scale Complex IT Systems. Development, Operation and Management: 17th Monterey Workshop 2012, Oxford, UK, March 19-21, 2012, Revised Selected Papers*. Hrsg. von R. Calinescu und D. Garlan. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, S. 183–208. ISBN: 978-3-642-34059-8. DOI: 10.1007/978-3-642-34059-8_10. URL: http://dx.doi.org/10.1007/978-3-642-34059-8_10.

[15] S. A. Hendrickson und A. van der Hoek. "Modeling Product Line Architectures Through Change Sets and Relationships". In: *Proceedings of the 29th International Conference on Software Engineering*. ICSE '07. Washington, DC, USA: IEEE Computer Society, 2007, S. 189–198. ISBN: 0-7695-2828-7. DOI: 10.1109/ICSE.2007.56. URL: http://dx.doi.org/10.1109/ICSE.2007.56.

[16] J. Kamischke, M. Lochau und H. Baller. "Conditioned Model Slicing of Feature-annotated State Machines". In: *Proceedings of the 4th International Workshop on Feature-Oriented Software Development*. FOSD '12. Dresden, Germany: ACM, 2012, S. 9–16. ISBN: 978-1-4503-1309-4. DOI: 10.1145/2377816.2377818. URL: http://doi.acm.org/10.1145/2377816.2377818.

[17] K. Kang, S. Cohen, J. Hess, W. Novak und A. Peterson. *Feature-Oriented Domain Analysis (FODA) Feasibility Study*. Techn. Ber. CMU/SEI-90-TR-021. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1990. URL: http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=11231.

[18] J. Kramer, J. Magee, M. Sloman und A. Lister. "CONIC: an integrated approach to distributed computer control systems". In: *Computers and Digital Techniques, IEE Proceedings E* 130.1 (1983), S. 1+. DOI: 10.1049/ip-e:19830001. URL: http://dx.doi.org/10.1049/ip-e:19830001.

[19] G. Lima, J. Santos, U. Kulesza, D. A. da Costa und S. V. Fialho. "A Delta Oriented Approach to the Evolution and Reconciliation of Enterprise Software Products Lines". In: *ICEIS 2013 - Proceedings of the 15th International Conference on Enterprise Information Systems, Volume 1, Angers, France, 4-7 July, 2013*. 2013, S. 255–263.

[20] S. Lity. "Konzeption und Evaluation eines delta-orientierten modellbasierten Testverfahrens für Softwareproduktlinien". Magisterarb. TU Braunschweig, 2011.

[21] S. Lity, M. Kowal und I. Schaefer. "Higher-Order Delta Modeling for Software Product Line Evolution". In: *Proceedings of the International Workshop on Feature-Oriented Software Development*. FOSD '16. to appear. Eindhoven, The Netherlands: ACM, 2016. ISBN: 978-1-4503-0208-1. DOI: 10.1145/1868688.1868696. URL: http://doi.acm.org/10.1145/1868688.1868696.

[22] S. Lity, R. Lachmann, M. Lochau und I. Schaefer. *Delta-oriented Software Product Line Test Models - The Body Comfort System Case Study*. Techn. Ber. 2012-07. Technische Universität Braunschweig, 2012.

[23] J. McGregor. *The Evolution of Product Line Assets*. Techn. Ber. CMU/SEI-2003-TR-005. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003. URL: http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=6601.

[24] T. Mens, A. Serebrenik und A. Cleve, Hrsg. *Evolving Software Systems*. Springer, 2014. ISBN: 978-3-642-45397-7. DOI: 10.1007/978-3-642-45398-4. URL: http://dx.doi.org/10.1007/978-3-642-45398-4.

[25] T. Müller, M. Lochau, S. Detering, F. Saust, H. Garbers, L. Märtin, T. Form und U. Goltz. *Comprehensive Description of a Model-based, Continuous Development Process for AUTOSAR Systems with Integrated Quality Assurance*. Techn. Ber. Technical Report 2009-06. Technische Universität Braunschweig, 2009. URL: http://www.digibib.tu-bs.de/?docid=00031645.

[26] L. Neves, P. Borba, V. Alves, L. Turnes, L. Teixeira, D. Sena und U. Kulesza. "Safe Evolution Templates for Software Product Lines". In: *J. Syst. Softw.* 106.C (08/2015), S. 42–58. ISSN: 0164-1212. DOI: 10.1016/j.jss.2015.04.024. URL: http://dx.doi.org/10.1016/j.jss.2015.04.024.

[27] M. Nieke, C. Seidl und S. Schuster. "Guaranteeing Configuration Validity in Evolving Software Product Lines". In: *Proceedings of the Tenth International Workshop on Variability Modelling of Software-intensive Systems*. VaMoS '16. Salvador, Brazil: ACM, 2016, S. 73–80. ISBN: 978-1-4503-4019-9. DOI: 10.1145/2866614.2866625. URL: http://doi.acm.org/10.1145/2866614.2866625.

[28] S. Oster, M. Zink, M. Lochau und M. Grechanik. "Pairwise Feature-interaction Testing for SPLs: Potentials and Limitations". In: *Proceedings of the 15th International Software Product Line Conference, Volume 2*. SPLC '11. Munich, Germany: ACM, 2011, 6:1–6:8. ISBN: 978-1-4503-0789-5. DOI: 10.1145/2019136.2019143. URL: http://doi.acm.org/10.1145/2019136.2019143.

[29] K. Pohl, G. Böckle und F. J. v. d. Linden. *Software Product Line Engineering: Foundations, Principles and Techniques*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005. ISBN: 3540243720.

[30] I. Schaefer. "Variability Modelling for Model-Driven Development of Software Product Lines". In: *VAMOS*. 2010.

[31] I. Schaefer, L. Bettini, F. Damiani und N. Tanzarella. "Delta-oriented Programming of Software Product Lines". In: *Proceedings of the 14th International Conference on Software Product Lines: Going Beyond*. SPLC'10. Jeju Island, South Korea: Springer-Verlag, 2010, S. 77–91. ISBN: 3-642-15578-2, 978-3-642-15578-9. URL: http://dl.acm.org/citation.cfm?id=1885639.1885647.

[32] I. Schaefer, R. Rabiser, D. Clarke, L. Bettini, D. Benavides, G. Botterweck, A. Pathak, S. Trujillo und K. Villela. "Software Diversity: State of the Art and Perspectives". In: *STTT* 14.5 (2012), S. 477–495. ISSN: 1433-2779.

[33] C. Seidl, I. Schaefer und U. Aßmann. “Integrated Management of Variability in Space and Time in Software Families”. In: *Proceedings of the 18th International Software Product Line Conference - Volume 1*. SPLC ’14. Florence, Italy: ACM, 2014, S. 22–31. ISBN: 978-1-4503-2740-4. DOI: 10.1145/2648511.2648514. URL: http://doi.acm.org/10.1145/2648511.2648514.

A Anhang

Im Folgenden ist der komplette Aufbau des Body Comfort Systems [22] als 150%-Modell abgebildet. Das 150%-Modell, welches ursprünglich von Oster et al. [28] modelliert wurde, wurde aus Lity et al. [22] entnommen. Das Body Comfort System beschreibt unterschiedliche Komfortfunktionen eines Fahrzeugs. Abb. A.1 zeigt dabei die Wurzel der State Machine, die sich in verschiedene Substate Machines aufgliedert. Diese Substate Machines zeigen das Verhalten für die Funktionen `Automatic Power Window` (Abb. A.2), `Manual Power Window` (Abb. A.3), `Finger Protection` (Abb. A.4), `Central Locking System` (Abb. A.5), `Safety Function` (Abb. A.6), `Exterior Mirror` (Abb. A.7), `Heatable` (Abb. A.8), `Alarm System` (Abb. A.9), `Control Automatic Power Window` (Abb. A.10), `Human Machine Interface` (Abb. A.11) sowie die verschiedenen LEDs (Abb. A.12, A.13, A.14, A.15, A.16, A.17, A.18 und A.19).

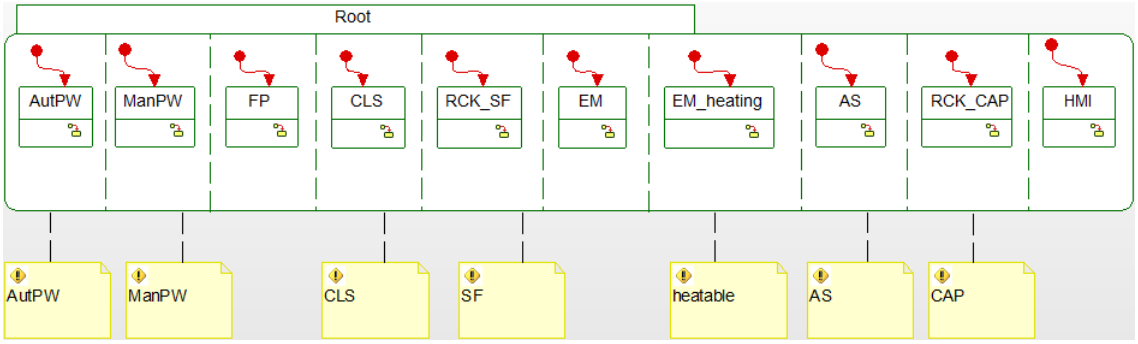

Abbildung A.1.: 150% State Machine `BCS Root` [22]

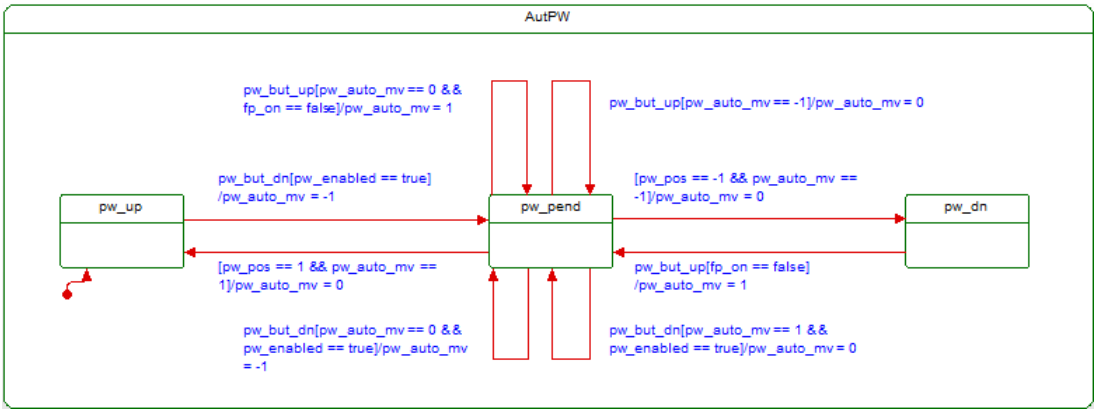

Abbildung A.2.: 150% Sub State Machine `AutPW` [22]

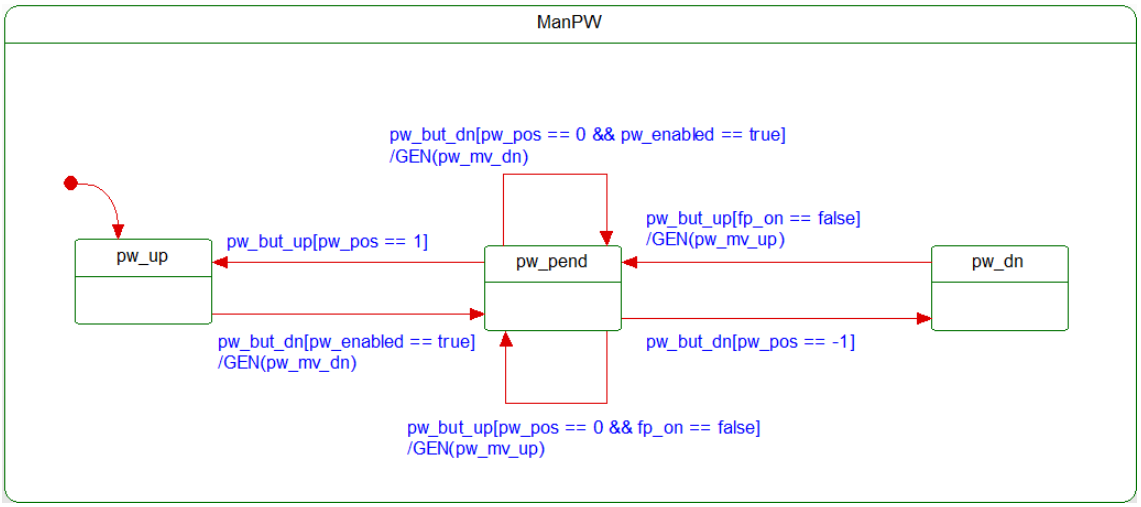

Abbildung A.3.: 150% Sub State Machine `ManPW` [22]

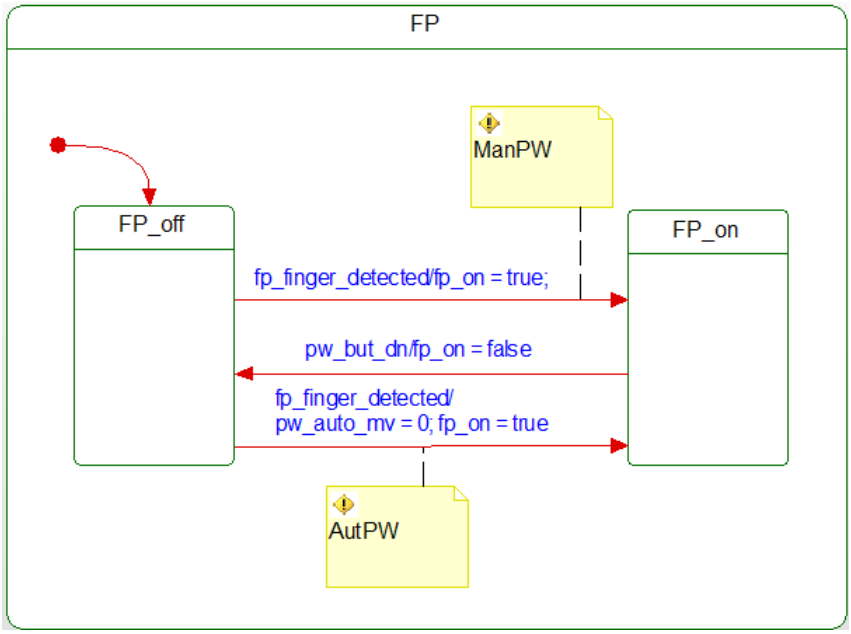

Abbildung A.4.: 150% Sub State Machine `FP` [22]

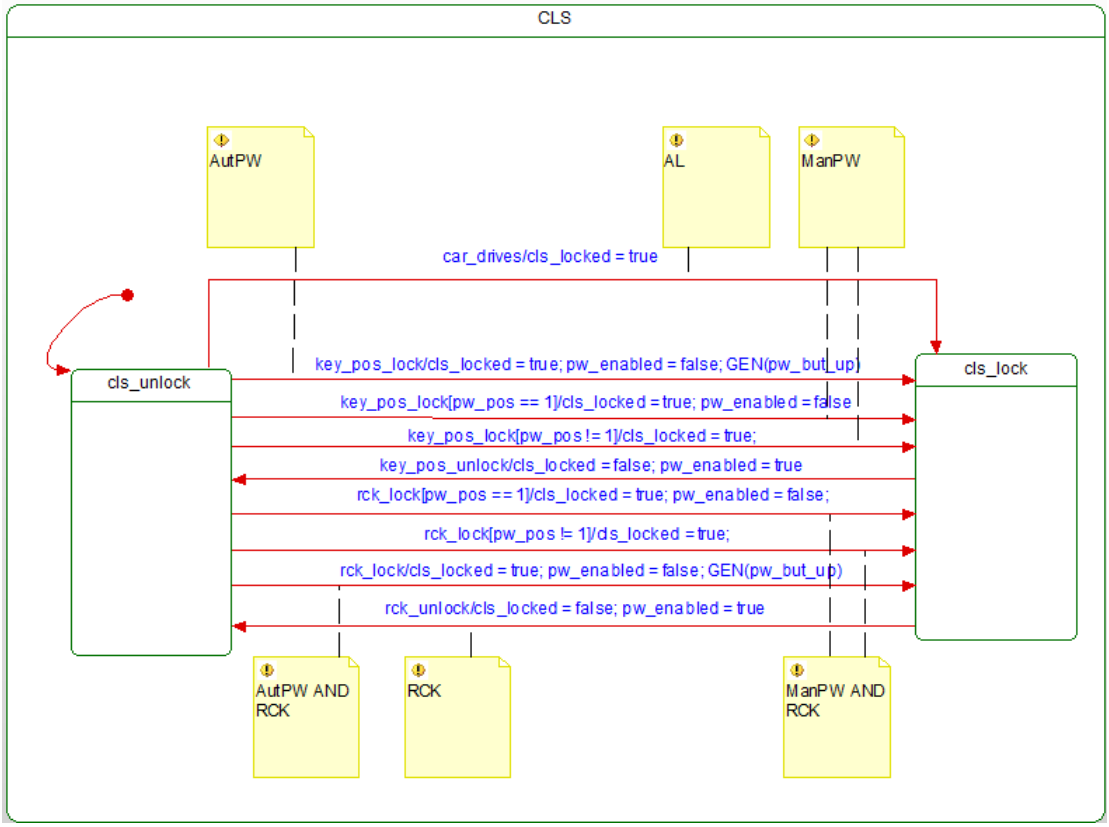

Abbildung A.5.: 150% Sub State Machine CLS [22]

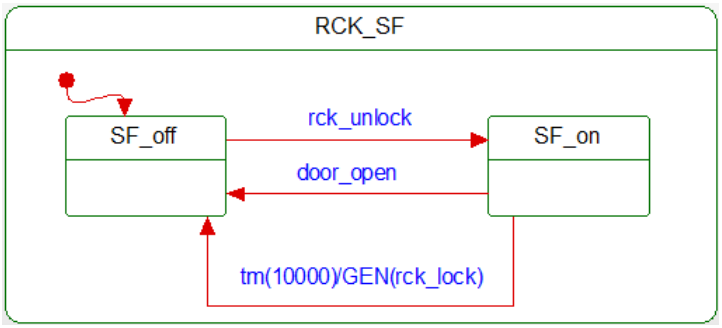

Abbildung A.6.: 150% Sub State Machine RCK_SF [22]

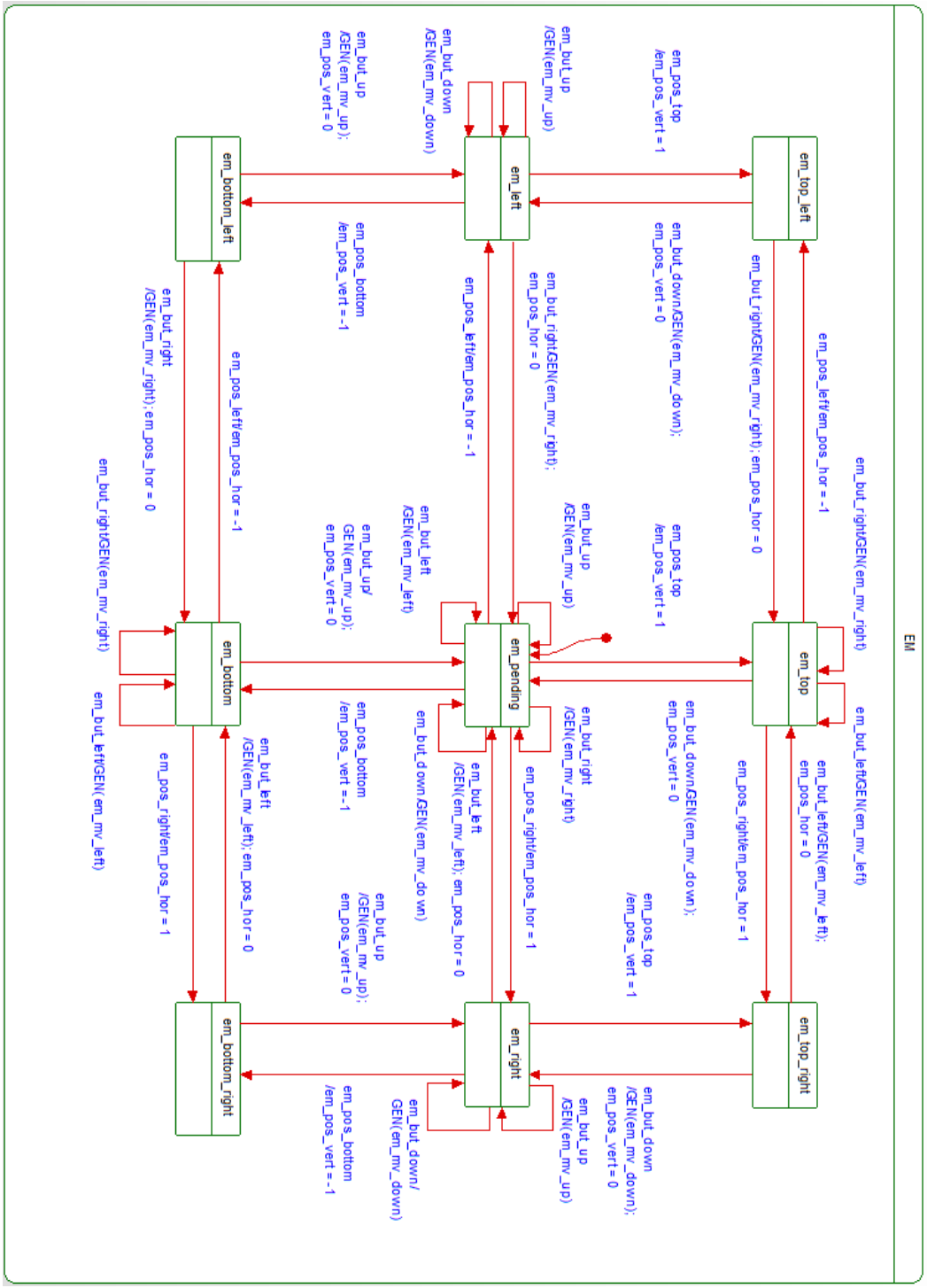

Abbildung A.7.: 150% Sub State Machine EM [22]

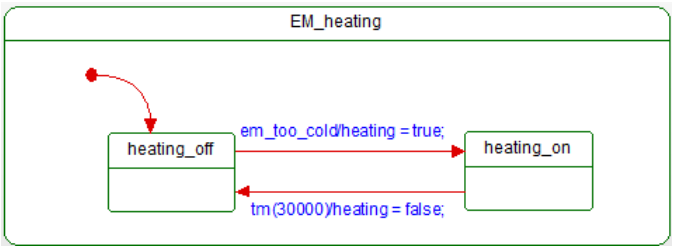

Abbildung A.8.: 150% Sub State Machine EM_heating [22]

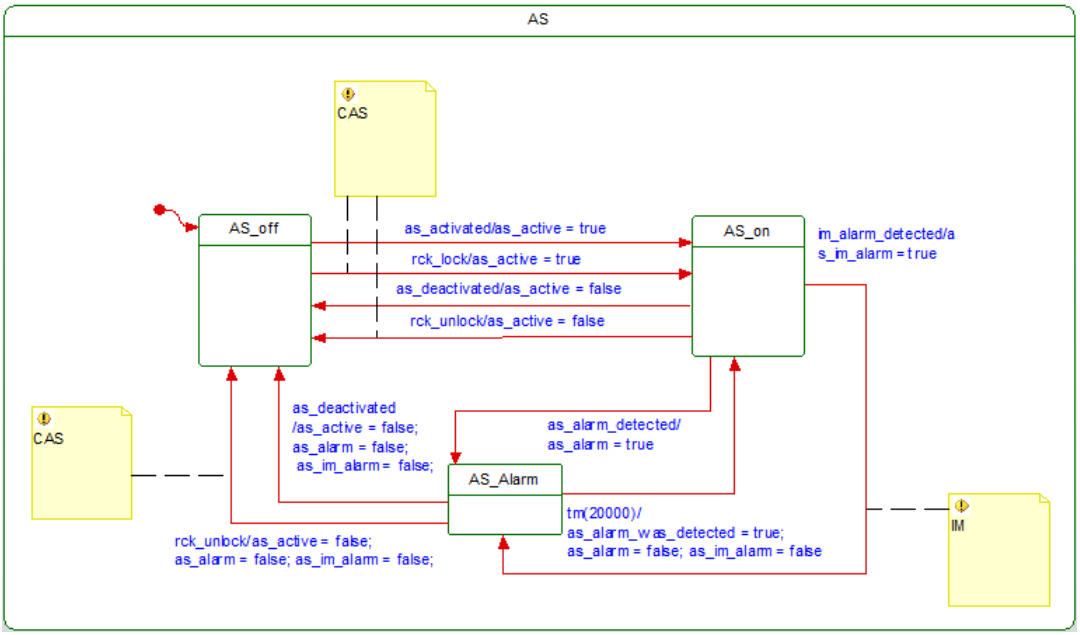

Abbildung A.9.: 150% Sub State Machine AS [22]

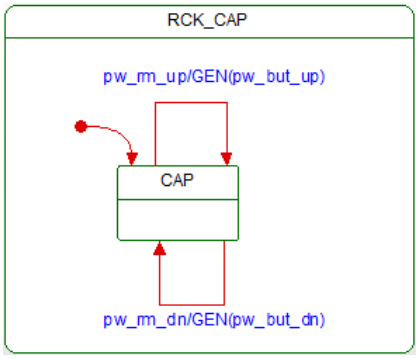

Abbildung A.10.: 150% Sub State Machine RCK_CAP [22]

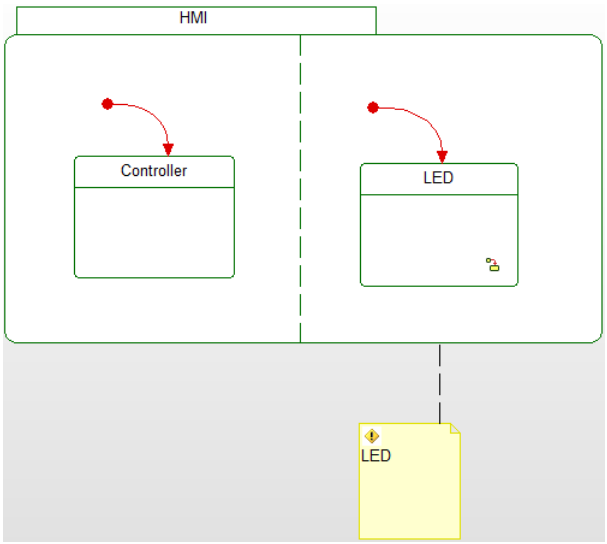

Abbildung A.11.: 150% Sub State Machine HMI [22]

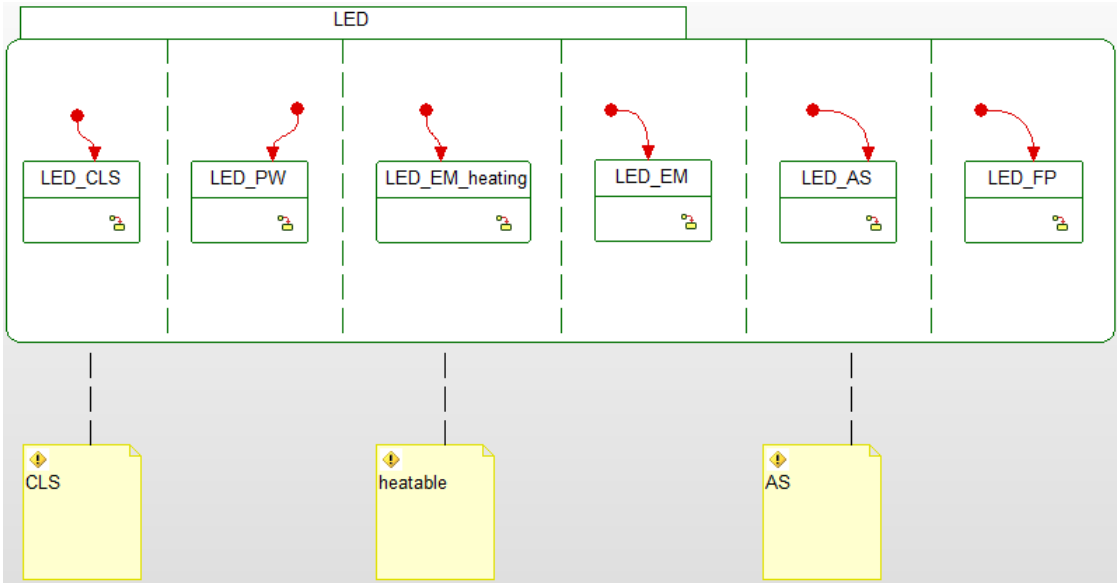

Abbildung A.12.: 150% Sub State Machine LED [22]

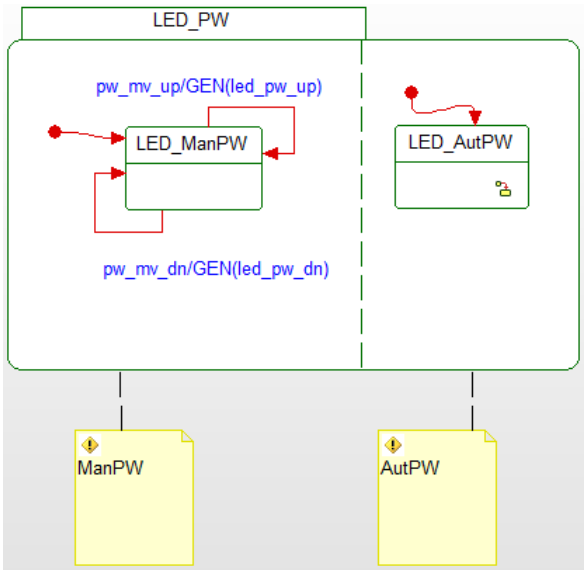

Abbildung A.13.: 150% Sub State Machine LED_PW [22]

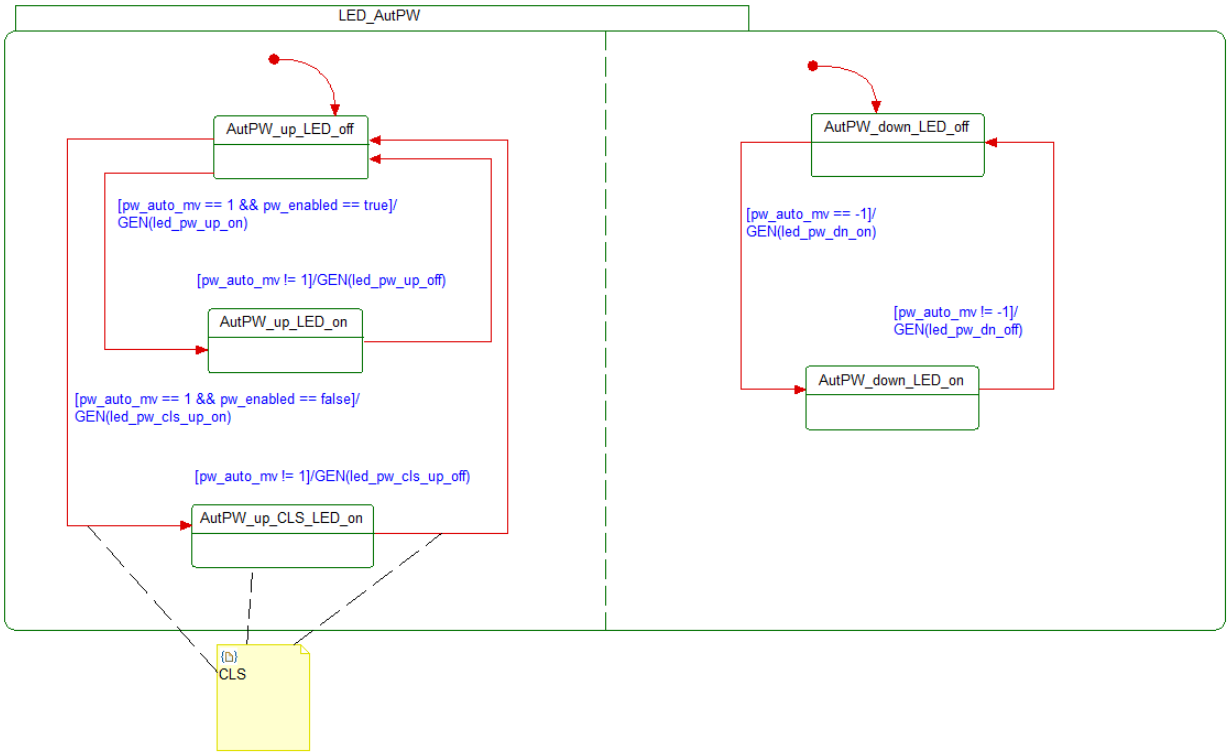

Abbildung A.14.: 150% Sub State Machine LED_AutPW [22]

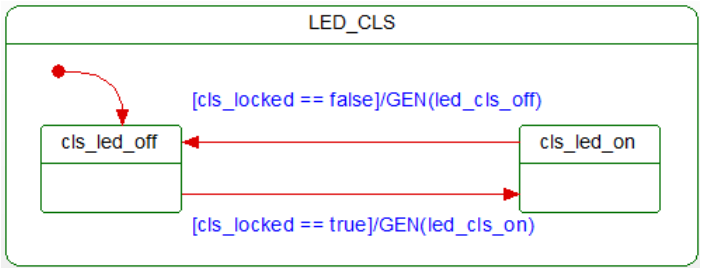

Abbildung A.15.: 150% Sub State Machine LED_CLS [22]

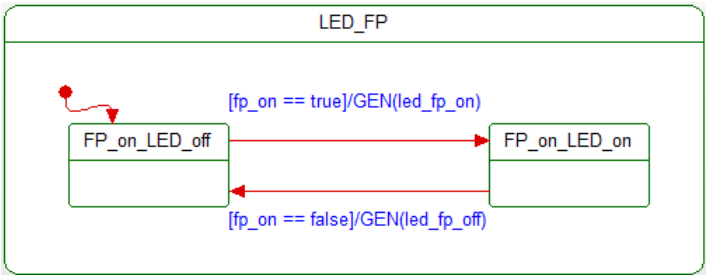

Abbildung A.16.: 150% Sub State Machine LED_FP [22]

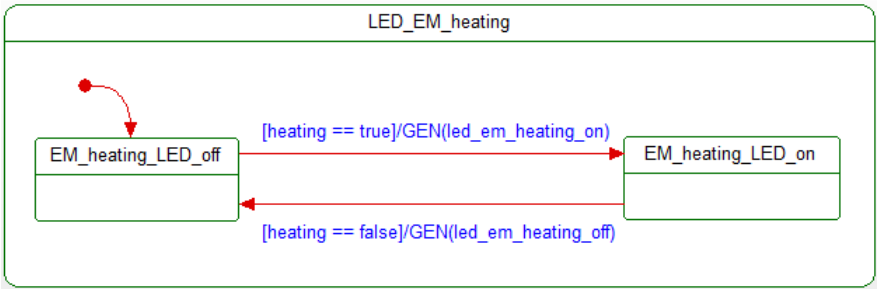

Abbildung A.17.: 150% Sub State Machine LED_EM_heating [22]

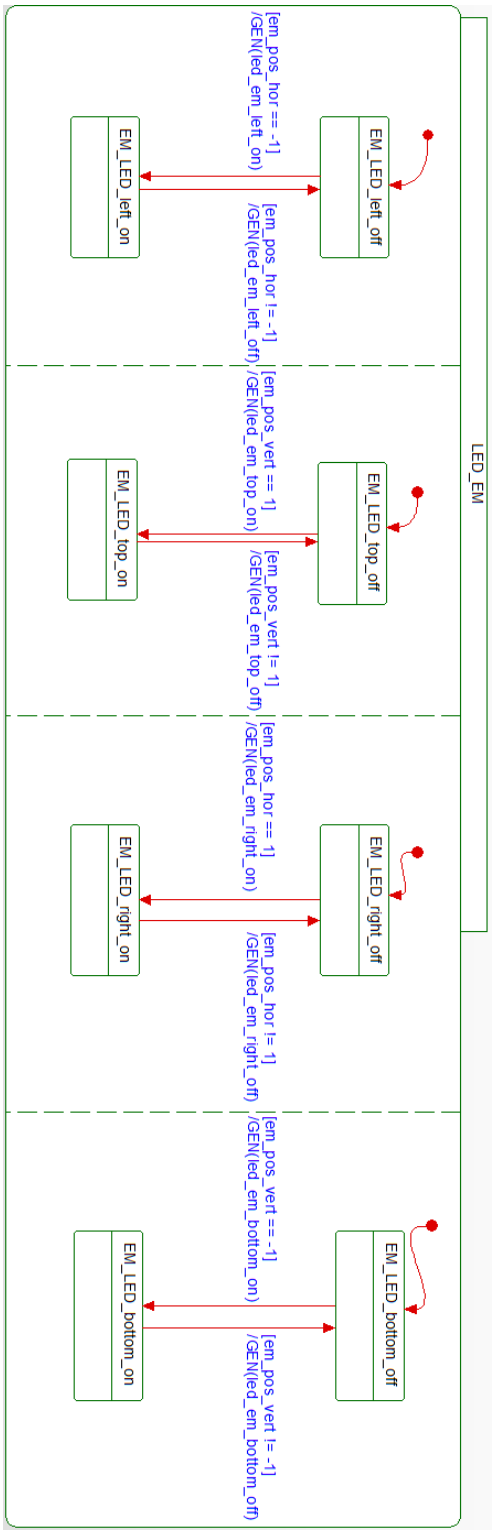

Abbildung A.18.: 150% Sub State Machine LED_EM [22]

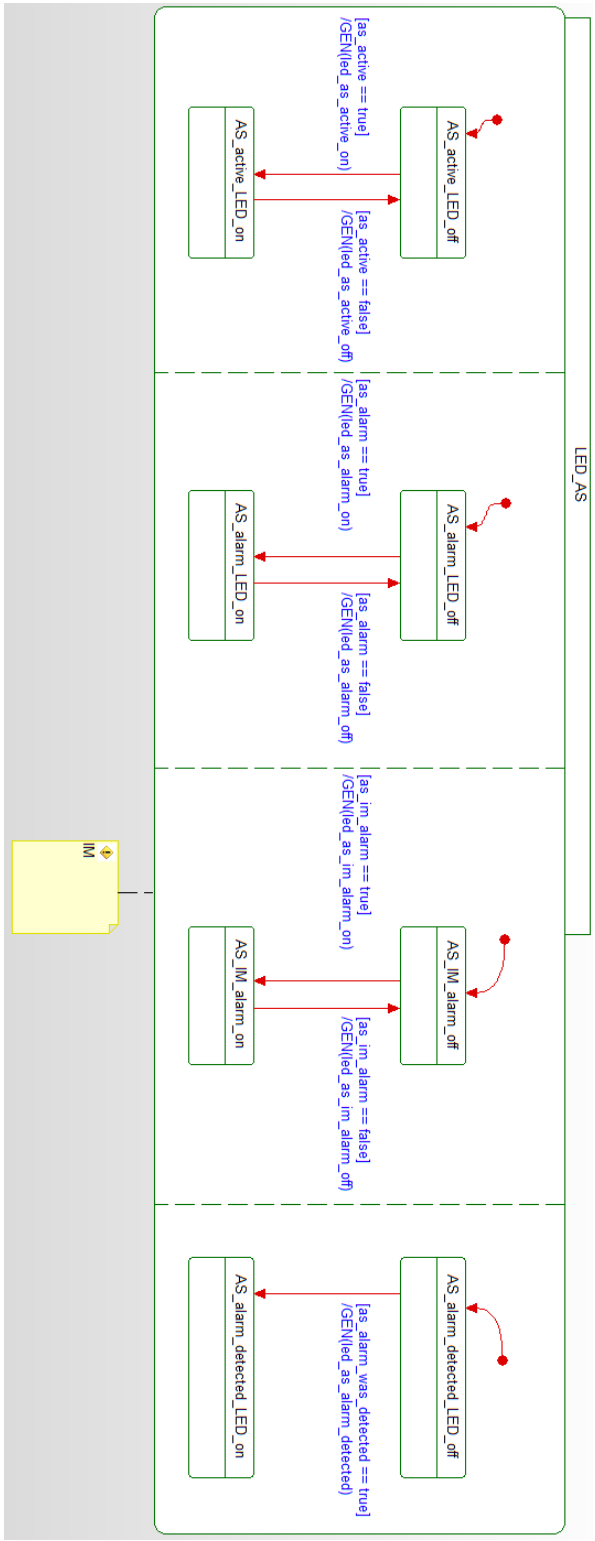

Abbildung A.19.: 150% Sub State Machine LED_AS [22]

Eidesstattliche Erklärung

Ich versichere, dass ich die beiliegende Projektarbeit ohne Hilfe Dritter und ohne Benutzung anderer, als der angegebenen Quellen und Hilfsmittel angefertigt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Diese Arbeit hat in gleicher Form noch keiner Prüfungsbehörde vorgelegen.
Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Ort, Datum Unterschrift

Technische Universität Carolo-Wilhelmina in Braunschweig
Institut für Softwaretechnik und Fahrzeuginformatik

Mühlenpfordtstr. 23
D-38106 Braunschweig